

BlackBerry Java SDK

UI and Navigation

Version: 6.0

Entwicklungshandbuch



Inhaltsverzeichnis

1	Erstellen einer Benutzeroberfläche, die mit den standardmäßigen BlackBerry-Benutzeroberflächen verträglich ist.....	6
2	BlackBerry-Gerät: Eingabe und Navigation für Benutzer.....	7
	Touchscreen.....	7
	Trackball oder Trackpad.....	9
	Trackball-Empfindlichkeit.....	9
	Trackball-Bewegung.....	9
	Trackwheel.....	9
	Tastatur.....	10
	Interaktionsmethoden auf BlackBerry-Geräten.....	10
3	Bildschirme.....	11
	Screen-Klassen.....	11
	Verwalten eines Zeichenbereichs.....	12
	Erstellen eines Bildschirmübergangs.....	13
	Codebeispiel: Erstellen eines Bildschirmübergangs.....	14
	Festlegen des Formats und der Richtung des Bildschirms.....	15
	Abrufen der Ausrichtung des Touchscreen.....	17
	Schränken Sie die Richtung des Touchscreens ein.....	17
	Benachrichtigen lassen, wenn sich die Größe des Zeichenbereichs auf dem Touchscreen ändert.....	17
4	Beschleunigungssensor.....	19
	Arten von Beschleunigungssensor-Daten.....	19
	Abrufen von Beschleunigungssensordaten.....	19
	Abrufen der Beschleunigungssensordaten in bestimmten Abständen.....	20
	Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Vordergrund befindet.....	21
	Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Hintergrund befindet.....	22
	Speichern der ausgelesenen Beschleunigungssensorwerte in einem Puffer.....	22
	Abrufen von Beschleunigungssensordaten aus einem Puffer.....	23
	Ermitteln des Zeitpunkts des Abrufens eines Werts vom Beschleunigungssensor.....	23
5	Ereignisse.....	24
	Reagieren auf Navigationsereignisse.....	24
	Bestimmen der Art der Eingabemethode.....	24
	Modelle für Touchscreen-Interaktionen.....	25
	Reagieren auf Touchscreen-Ereignisse.....	26
	Antworten auf Systemereignisse, während der Benutzer den Bildschirm berührt.....	27

Reagieren auf einen Benutzer, der einen Finger schnell nach oben über den Bildschirm bewegt.....	27
Reagieren auf einen Benutzer, der einen Finger schnell nach unten über den Bildschirm bewegt.....	28
Reagieren auf einen Benutzer, der einen Finger schnell nach rechts über den Bildschirm bewegt.....	29
Reagieren auf einen Benutzer, der schnell einen Finger nach rechts über den Bildschirm bewegt.....	30
Reagieren auf einen Benutzer, der auf den Bildschirm klickt.....	30
Reagieren auf einen Benutzer, der den Bildschirm zweimal schnell berührt.....	31
Reagieren auf einen Benutzer, der ein Element auf dem Bildschirm berührt und zieht.....	32
Reagieren auf einen Benutzer, der den Bildschirm leicht berührt.....	32
Reagieren auf eine Bildlaufaktion.....	33
Reagieren auf einen Benutzer, der den Bildschirm an zwei Stellen gleichzeitig berührt.....	33
Pinch-Gesten.....	34
Aktivieren von Pinch-Gesten.....	35
Fingerstrich-Gesten mit Trackpads.....	36
Aktivieren von Fingerstrich-Gesten, die Benutzer auf dem Trackpad machen.....	36
Ereignisinjektion.....	37
6 Command Framework API.....	38
Verwenden eines Befehls mit einer Benutzeroberflächenkomponente.....	39
Verwenden eines Befehl in mindestens einer Anwendung.....	40
7 Anordnen von Benutzeroberflächenkomponenten.....	43
Anordnen von Benutzeroberflächenkomponenten.....	44
Erstellen eines Raster-Layouts.....	44
Erstellen einer Gittergröße.....	46
Anzeigen von Feldern in einem Paar temporärer Manager.....	48
Vorübergehendes Anzeigen eines ButtonField und eines LabelField oben bzw. unten auf dem Bildschirm.....	48
Anzeigen eines Felds an einer absoluten Position auf dem Bildschirm.....	51
Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm.....	51
Codebeispiel: Anzeigen einer Bezeichnung an einer absoluten Bildschirmposition.....	52
8 Benutzeroberflächen-Komponenten.....	53
Hinzufügen einer Benutzeroberflächenkomponente zu einem Bildschirm.....	53
Ausrichten eines Feldes an einer Textzeile.....	53
Schaltflächen.....	53
Bewährtes Verfahren: Implementieren von Schaltflächen.....	54
Erstellen einer Schaltfläche.....	55
Kontrollkästchen.....	55
Bewährtes Verfahren: Implementieren von Kontrollkästchen.....	55

Erstellen eines Kontrollkästchens.....	56
Erstellen einer Bitmap.....	58
Erstellen eines benutzerdefinierten Feldes.....	58
Erstellen eines Felds, um Webinhalte anzuzeigen.....	61
Anzeigen von HTML-Inhalten in einem Browserfeld.....	61
Anzeigen von HTML-Inhalten von einer Webseite in einem Browserfeld.....	63
Anzeigen von HTML-Inhalten aus einer Ressource in Ihrer Anwendung.....	64
Konfigurieren eines Browserfeldes.....	66
Senden von Formulardaten an eine Webadresse in einem Browserfeld.....	67
Dialogfelder.....	69
Bewährtes Verfahren: Implementieren von Dialogfeldern.....	70
Erstellen eines Dialogfelds.....	72
Dropdown-Listen.....	72
Bewährtes Verfahren: Implementieren von Dropdown-Listen.....	73
Erstellen einer Dropdown-Liste.....	74
Bezeichnungen.....	76
Bewährtes Verfahren: Implementieren von Bezeichnungen.....	76
Erstellen eines Textetiketts.....	76
Listen und Tabellen.....	77
Bewährtes Verfahren: Implementieren von Listen und Tabellen.....	79
Erstellen eines Listenfelds.....	80
Optionsfelder.....	82
Bewährtes Verfahren: Implementieren von Optionsfeldern.....	82
Erstellen eines Optionsfeldes.....	83
Aktivitäts- und Fortschrittsanzeigen.....	85
Bewährtes Verfahren: Implementieren von Aktivitäts- und Fortschrittsanzeigen.....	86
Anzeigen von Aktivität oder Aufgabenfortschritt.....	87
Auswahltool.....	94
Bewährtes Verfahren: Implementieren von Auswahlen.....	96
Erstellen einer Datumsauswahl.....	96
Erstellen einer Dateiauswahl.....	98
Suche.....	101
Bewährtes Verfahren: Implementieren von Suchfunktionen.....	103
Erstellen eines Suchfelds.....	104
Autocomplete-Textfeld.....	106
Drehfelder.....	111
Erstellen eines Drehfelds.....	112
Bewährtes Verfahren: Implementieren von Drehfeldern.....	114
Textfelder.....	115

Bewährtes Verfahren: Implementieren von Textfeldern.....	116
Erstellen eines Textfelds.....	117
Erstellen eines Datumsfelds.....	117
Erstellen eines Kennwortfelds.....	118
Strukturansichten.....	119
Bewährtes Verfahren: Implementieren von Strukturansichten.....	119
Erstellen eines Feldes für das Anzeigen einer Strukturansicht.....	120
9 Bilder.....	121
Verwenden von codierten Bildern.....	121
Zugriff auf ein codiertes Bild über einen Eingabestream.....	121
Codieren eines Bilds.....	121
Anzeigen eines codierten Bilds.....	122
Festlegen der Anzeigegröße eines codierten Bilds.....	122
Festlegen des Decodierungsmodus für ein Bild.....	122
Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken.....	122
Codebeispiel: Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken.....	123
Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken.....	123
Anzeigen mehrerer Bilder zum Scrollen.....	124
Codebeispiel: Anzeigen einer Reihe von Bildern zum Scrollen.....	124
Anzeigen mehrerer Bilder zum Scrollen.....	125
10 Menübefehle.....	128
Erstellen eines Menüs.....	128
Codebeispiel: Erstellen eines Menüs.....	129
Bewährtes Verfahren: Implementieren von Menüs.....	130
Untermenüs.....	130
Bewährtes Verfahren: Implementieren von Untermenüs.....	132
Erstellen eines Untermenüs.....	133
Codebeispiel: Erstellen eines Untermenüs.....	134
Popup-Menüs.....	135
Bewährtes Verfahren: Implementieren von Popup-Menüs.....	136
Info über das Einfügen von Elementen in Popup-Menüs.....	137
Unterstützung für ältere Kontextmenüs.....	138
Erstellen eines Popup-Menüs.....	138
Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software.....	143
Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung.....	144
Ändern der Darstellung eines Menüs.....	145
Ändern der Darstellung eines Menüs.....	145

Codebeispiel: Ändern der Darstellung eines Menüs.....	147
11 Benutzerdefinierte Schriftarten.....	149
Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung.....	149
Codebeispiel: Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry Java-Anwendung.....	150
12 Rechtschreibprüfung.....	152
Hinzufügen einer Funktion zur Rechtschreibprüfung.....	152
Empfangen von Rechtschreibprüfungsereignissen.....	153
13 Verwandte Ressourcen.....	154
14 Glossar.....	155
15 Rückmeldung.....	156
16 Verlauf der Dokumentrevision.....	157
17 Rechtliche Hinweise.....	159

Erstellen einer Benutzeroberfläche, die mit den standardmäßigen BlackBerry-Benutzeroberflächen verträglich ist

1

Mithilfe der Standard-MIDP-APIs und der BlackBerry®-Benutzeroberflächen-APIs können Sie eine Benutzeroberfläche für die BlackBerry® Java® Application erstellen.

Die Benutzeroberflächen-Komponenten in den BlackBerry-Benutzeroberflächen-APIs wurden dafür entworfen, Layouts und Verhaltensweisen bereitzustellen, die den BlackBerry® Device Software-Anwendungen entsprechen. Wenn Sie diese APIs nutzen, können Sie zum einen Komponenten verwenden, mit denen BlackBerry-Benutzer vertraut sind, und zum anderen Zeit sparen, da Sie diese Komponenten nicht selbst erstellen müssen.

- Bildschirmkomponenten können ein Standardbildschirmlayout, ein Standardmenü und ein Standardverhalten bereitstellen, wenn ein BlackBerry-Gerätebenutzer die Esc-Taste drückt, mit dem Trackpad klickt oder den Touchscreen berührt.
- Feldkomponenten können die standardmäßigen Benutzeroberflächenelemente für die Datumsauswahl, Optionsschaltflächen, Kontrollkästchen, Listen, Textfelder, Bezeichnungen und Fortschrittsanzeigen bereitstellen.
- Layout-Manager können einer Anwendung die Möglichkeit geben, die Komponenten auf standardmäßige Weise auf einem BlackBerry-Gerätebildschirm anzuordnen, z. B. horizontal, vertikal oder von links nach rechts.

Mithilfe der BlackBerry-Benutzeroberflächen-APIs können Sie eine Benutzeroberfläche mit Tabellen, Rastern und anderen speziellen Funktionen erstellen. Sie können ein standardmäßiges Java-Ereignismodell verwenden, um bestimmte Ereignisarten zu empfangen und auf diese zu antworten. Eine BlackBerry-Geräteanwendung kann Benutzerereignisse, z. B. wenn der Benutzer mit dem Trackpad klickt oder Eingaben über die Tastatur vornimmt, sowie Systemereignisse, z. B. globale Warnungen, Uhrzeitänderungen und USB-Anschlussverbindungen, empfangen und auf sie antworten.

BlackBerry-Gerät: Eingabe und Navigation für Benutzer

2

BlackBerry®-Geräte umfassen eine Tastatur, ein Trackwheel, Trackpad oder einen Trackball sowie eine Esc-Taste für die Eingabe und die Navigation. Auf BlackBerry-Geräten mit einem SurePress™-Touchscreen entspricht das Klicken auf dem Bildschirm dem Klicken mit dem Trackball oder Trackwheel.

Eine BlackBerry® Java® Application sollte weitestgehendst das folgende Eingabe- und Navigationsmodell verwenden.

- Klicken auf das Trackwheel, Trackpad, den Trackball oder Touchscreen öffnet in der Regel ein Menü.
- Durch Drücken der Esc-Taste können Aktionen abgebrochen werden. Weiterhin ist damit die Rückkehr zum vorherigen Bildschirm möglich. Durch Drücken der Esc-Taste werden Benutzer immer wieder zur Startseite zurückkehren. Durch Halten der Esc-Taste wird der Browser oder die Medienanwendung geschlossen.

Standardmäßig bieten die BlackBerry-Bildschirmobjekte diese Funktionalität ohne weiteres Anpassen. Sie müssen allerdings die gewünschten Menüeinträge und die zusätzliche Benutzeroberflächen- und Navigationslogik hinzufügen.

Touchscreen

Bei BlackBerry®-Geräten mit einem SurePress™-Touchscreen verwenden Benutzer einen Finger, um mit den Anwendungen auf dem Gerät zu interagieren. Durch das Ausführen verschiedener Aktionen auf dem Touchscreen geben Benutzer Text ein und navigieren durch die Bildschirme.

Benutzer können auch Aktionen durch Klicken auf Symbole auf der Shortcut-Leiste oder durch Drücken der Menütaste ausführen.

Bei BlackBerry-Geräten mit einem Touchscreen können Benutzer die folgenden Aktionen ausführen:

Aktion	Ergebnis
Leichtes Berühren des Bildschirms	Diese Aktion markiert ein Element. Wenn Benutzer in einem Textfeld den Bildschirm in der Nähe des Cursors berühren, wird ein umrandetes Feld um den Cursor angezeigt. Dieses Feld hilft Benutzern, den Cursor problemloser zu verschieben.
Tippen auf den Bildschirm	In Anwendungen, die eine Vollbildanzeige unterstützen, wie z. B. BlackBerry® Maps und BlackBerry® Browser, wird durch diese Aktion die Shortcut-Leiste verborgen bzw. angezeigt.
Zweimaliges Tippen auf den Bildschirm	In einer Webseite, Karte, Bild oder Präsentationsanlage zoomt diese Aktion in die jeweilige Webseite, Karte, Bild oder Präsentationsanlage.
Den Finger auf einem Element halten	Durch diese Aktion wird auf der Shortcut-Leiste ein QuickInfo angezeigt, das die Aktion beschreibt, die das Symbol darstellt.

Aktion	Ergebnis
	Wenn Benutzer in einer Nachrichtenliste einen Finger auf den Absender oder den Betreff einer Nachricht halten, sucht das BlackBerry-Gerät nach dem Absender oder dem Betreff.
Berühren und Ziehen eines Elements auf dem Bildschirm	Diese Aktion bewegt den Inhalt auf dem Bildschirm in die entsprechende Richtung. Wenn Benutzer zum Beispiel ein Menüelement berühren und ziehen, bewegt sich die Liste der Menüelemente in die gleiche Richtung. In einem Textfeld bewegt diese Aktion das umrandete Feld und den Cursor in die gleiche Richtung.
Berühren des Bildschirms an zwei Stellen gleichzeitig	Diese Aktion markiert den Text oder die Liste der Elemente, wie z. B. Nachrichten, zwischen den beiden Stellen. Zum Hinzufügen oder Entfernen markierter Texte oder Elemente können Benutzer den Bildschirm an einer anderen Stelle berühren.
Klicken (Drücken) auf den Bildschirm	Leitet eine Aktion ein. Wenn Benutzer zum Beispiel auf ein Element in einer Liste klicken, wird der mit dem Element verknüpfte Bildschirm angezeigt. Diese Aktion entspricht dem Klicken auf das Trackwheel, Trackpad oder den Trackball. In einer Karte, Bild oder Präsentationsanlage zoomt diese Aktion in die jeweilige Karte, Bild oder Präsentationsanlage. In einer Webseite zoomt diese Aktion in die Webseite oder folgt einem Link. In einem Textfeld wird durch diese Aktion der Cursor positioniert. Wenn das Feld Text enthält, erscheint ein umrandetes Feld um den Cursor.
Schnelles Fahren mit einem Finger nach oben oder unten über den Bildschirm	Durch schnelles Fahren mit einem Finger nach oben wird der nächste Bildschirm angezeigt. Durch schnelles Fahren mit einem Finger nach unten wird der vorherige Bildschirm angezeigt. Wenn die Tastatur angezeigt wird, wird durch schnelles Fahren mit einem Finger nach unten die Tastatur verborgen und die Shortcut-Leiste angezeigt.
Schnelles Fahren mit einem Finger nach links oder rechts über den Bildschirm	Diese Aktion zeigt das nächste bzw. vorherige Bild oder Nachricht, oder den nächsten bzw. vorherigen Tag, Woche oder Monat in einem Kalender an.
Schieben mit einem Finger nach oben oder unten über den Bildschirm	In der Kamera, wird durch Schieben eines Fingers ein Objekt herangezoomt. Durch Schieben mit einem Finger nach unten wird aus dem Objekt herausgezoomt.
Schieben eines Fingers in irgendeine Richtung	Diese Aktion führt eine Karte oder Webseite nach. Wenn Benutzer ein Bild heranzoomen, wird durch diese Aktion auch ein Bild nachgeführt.
Drücken der Esc-Taste	Diese Aktion entfernt die Markierung von Text oder einer Liste von Elementen. In einer Webseite, Karte oder einem Bild zoomt diese Aktion um eine Ebene heraus. Benutzer können die Esc-Taste zweimal drücken, um zur ursprünglichen Ansicht zurück zu zoomen.

Trackball oder Trackpad

Auf BlackBerry®-Geräten mit einem Trackball oder Trackpad dienen der Trackball oder das Trackpad als primäre Steuerung der Benutzernavigation. Benutzer können folgende Aktionen ausführen:

- Drehen des Trackballs oder Bewegen eines Fingers auf dem Trackpad, um den Cursor zu bewegen.
- Klicken mit dem Trackball oder Trackpad, um Standardaktionen auszuführen oder ein Kontextmenü zu öffnen.
- Klicken mit dem Trackball oder Trackpad bei gleichzeitigem Drücken der Umschalttaste, um Text zu markieren oder Nachrichten in einer Nachrichtenliste zu markieren.

BlackBerry mit einem Trackball oder Trackpad verfügen auch über eine Menütaste, die sich links vom Trackball bzw. dem Trackpad befindet. Benutzer können die Menütaste drücken, um ein vollständiges Menü verfügbarer Aktionen zu öffnen.

Trackball-Empfindlichkeit

Die Trackball-Empfindlichkeit bezeichnet den Umfang der Trackball-Bewegung, die erforderlich ist, damit das System die Bewegung als Navigationsereignis erkennt und entsprechend ein Navigationsereignis an die Softwareschicht zu senden. Die BlackBerry®-Gerätehardware misst die physische Trackball-Bewegung mithilfe spezieller Einheiten, die als Ticks bezeichnet werden. Wenn die Anzahl der Ticks entlang einer Achse den Schwellenwert des Systems oder einer BlackBerry-Geräteanwendung überschreitet, wird ein Navigationsereignis an die Softwareschicht gesendet, und das System setzt die Tick-Zahl wieder auf Null zurück. Auch nach einer bestimmten Leerlaufzeit wird die Tick-Zahl wieder auf Null zurückgesetzt.

Mit der Trackball-API können Sie die Trackball-Empfindlichkeit festlegen. Eine hohe Trackball-Empfindlichkeit entspricht einem kleineren Tick-Schwellenwert (ein geringe Trackball-Bewegung erzeugt ein Navigationsereignis). Eine niedrige Trackball-Empfindlichkeit entspricht einem größeren Tick-Schwellenwert (ein größere Trackball-Bewegung ist erforderlich, um ein Navigationsereignis zu erzeugen).

Trackball-Bewegung

Mit der Trackball-API filtern Sie die Trackball-Bewegungsdaten, die die Hardware des BlackBerry®-Geräts an die Softwareschicht sendet. Die Trackball-API filtert Bewegungen "Rauschen" oder unerwünschte Bewegungen.

Mit der Trackball-API können Sie auch Einstellungen ändern, z. B. die Beschleunigung beim Bewegen des Trackballs. Eine höhere Einstellung für die Beschleunigung beim Bewegen des Trackballs kann dazu führen, dass die Softwareschicht die Bewegungen als scheinbar schneller erkennt als die BlackBerry-Gerätehardware, solange der Benutzer den Trackball fortlaufend rollt. Die Trackball-Empfindlichkeit steigt vorübergehend an, wenn der Benutzer den Trackball längere Zeit ununterbrochen rollt.

Trackwheel

BlackBerry®-Geräte vor der BlackBerry® Pearl™ 8100 Series verwenden ein Trackwheel als primäre Steuerung der Benutzernavigation. Das Trackwheel befindet sich an der rechten Seite des BlackBerry® -Geräts.

Benutzer können folgende Aktionen ausführen:

- Trackwheel rollen, um den Cursor in vertikaler Richtung zu bewegen
- Trackwheel rollen und gleichzeitig die Alt-Taste drücken, um den Cursor in horizontaler Richtung zu bewegen
- Mit dem Trackwheel klicken, um eine Aktion einzuleiten oder das Menü zu öffnen

Tastatur

Benutzer verwenden die Tastatur hauptsächlich, um Text einzugeben. Buchstabentasten senden dem BlackBerry®-Gerät ein Zeichen. Eine Zusatztaste verändert die Funktion von Zeichentasten. Zu den Zusatztasten gehören die Umschalttaste und die Alt-Taste. Wenn Benutzer eine Zusatztaste drücken, wird eine Texteingabemodusanzeige in der rechten oberen Ecke des Bildschirms angezeigt.

Auf BlackBerry ohne Touchscreen können Benutzer auch die Tastatur verwenden, um auf dem Bildschirm zu navigieren (zum Beispiel zum Navigieren in einer Karte). Die Navigation mithilfe der Tastatur sollte jedoch immer eine Alternative zur Navigation mithilfe von Trackwheel, Trackball oder Trackpad sein.

BlackBerry-Geräte haben entweder eine komplette Tastatur oder eine reduzierte Tastatur. Auf BlackBerry-Geräten mit einem SurePress™-Touchscreen wird in den meisten Fällen die komplette Tastatur im Querformat und die reduzierte Tastatur im Hochformat angezeigt.

Interaktionsmethoden auf BlackBerry-Geräten

BlackBerry-Gerätemodell	Interaktionsmethode
BlackBerry® 7100 Series	Trackwheel
BlackBerry® 8700 Series	Trackwheel
BlackBerry® 8800 Series	Trackball
BlackBerry® Bold™ 9000-Smartphone	Trackball
BlackBerry® Bold™ 9650-Smartphone	Trackpad
BlackBerry® Bold™ 9700-Smartphone	
BlackBerry® Curve™ 8300 Series	Trackball
BlackBerry® Curve™ 8500 Series	Trackpad
BlackBerry® Curve™ 8900-Smartphone	Trackball
BlackBerry® Pearl™ 8100 Series	Trackball
BlackBerry® Pearl™ Flip 8200 Series	Trackball
BlackBerry® Storm™ 9500 Series	Touchscreen
BlackBerry® Tour™ 9630-Smartphone	Trackball

Bildschirme

3

Die Hauptkomponente einer Benutzeroberfläche des BlackBerry®-Geräts ist das `Screen`-Objekt.

Auf einem BlackBerry-Gerät können mehrere Bildschirme gleichzeitig geöffnet sein, aber nur ein Bildschirm zur selben Zeit angezeigt werden. Die BlackBerry® Java® Virtual Machine verwaltet einen sortierten Satz von `Screen`-Objekten in einem Anzeige-Stapel. Der Bildschirm oben im Stapel ist der aktive Bildschirm, der für den BlackBerry-Gerätebenutzer sichtbar ist. Wenn eine BlackBerry-Geräteanwendung einen Bildschirm anzeigt, wird der Bildschirm im Stapel nach oben geschoben. Wenn eine BlackBerry-Geräteanwendung einen Bildschirm schließt, verschwindet der Bildschirm vom Stapel, und der nächste Bildschirm im Stapel wird ausgegeben (und gegebenenfalls neu gezeichnet).

Jeder Bildschirm kann nur einmal im Anzeige-Stapel vorkommen. Die BlackBerry JVM löst einen Laufzeitfehler aus, wenn ein Bildschirm, den die BlackBerry-Geräteanwendung auf den Stapel schiebt, bereits existiert. Eine BlackBerry-Geräteanwendung muss einen Bildschirm vom Anzeige-Stapel nehmen, wenn der Benutzer die Interaktion mit dem Bildschirm beendet, damit die BlackBerry-Geräteanwendung den Speicher effizient einsetzen kann. Sie sollten nur ein paar modale Bildschirme gleichzeitig verwenden, denn jeder Bildschirm verwendet einen separaten Thread.

Die Benutzeroberflächen-APIs initialisieren einfache `Screen`-Objekte. Nachdem Sie einen Bildschirm erstellen, können Sie Felder und ein Menü hinzufügen, und den Bildschirm dem Benutzer anzeigen, indem Sie ihn im Anzeige-Stapel nach oben schieben. Sie können die Benutzeroberfläche des BlackBerry-Geräts ändern und bei Bedarf neue Feldtypen implementieren. Sie können auch benutzerdefinierte Navigation hinzufügen.

Die `Screen`-Klasse implementiert keine Disambiguierung, die für komplexe Eingabemethoden, wie z. B. internationale Tastaturen, erforderlich ist. Für die Integration der verschiedenen Eingabemethoden können Sie die `Field`-Klasse oder eine ihrer Unterklassen erweitern. Sie sollten `Screen`-Objekte nicht für die Texteingabe verwenden.

Screen-Klassen

Class	Beschreibung
<code>Screen</code>	Sie können die <code>Screen</code> -Klasse verwenden, um einen Bildschirm zu erstellen, und einen Layout-Manager verwenden, um die Benutzeroberflächen-Komponenten auf dem Bildschirm anzulegen. Sie können eine bestimmte Art von Bildschirm mithilfe der Stile definieren, die von den Konstanten in der übergeordneten <code>Field</code> -Klasse definiert werden.
<code>FullScreen</code>	Sie können die <code>FullScreen</code> -Klasse verwenden, um einen leeren Bildschirm zu erstellen, dem Sie Benutzeroberflächen-Komponenten in einem standardmäßigen vertikalen Layout hinzufügen können. Wenn Sie einen anderen Layoutstil verwenden wollen, wie z. B. horizontal oder diagonal, können Sie stattdessen die <code>Screen</code> -Klasse verwenden und einen Layout-Manager hinzufügen.

Class	Beschreibung
MainScreen	Sie können die MainScreen-Klasse verwenden, um einen Bildschirm mit den folgenden standardmäßigen Benutzeroberflächen-Komponenten zu erstellen: • Standardbildschirmtitel, mit einem SeparatorField nach dem Titel • Bildlauffähiger Hauptabschnitt, in einem VerticalFieldManager enthalten • Standardmenü mit einem Menüelement "Schließen" • Standardmäßige Schließen-Aktion, die den Bildschirm schließt, wenn der BlackBerry®-Gerätebenutzer auf das Menüelement "Schließen" klickt oder die Esc-Taste drückt Sie sollten den Einsatz eines MainScreen-Objekts für den ersten Bildschirm Ihrer BlackBerry-Geräteanwendung in Betracht ziehen, um die Konsistenz mit anderen BlackBerry-Geräteanwendungen beizubehalten.

Verwalten eines Zeichenbereichs

Das Graphics-Objekt stellt die gesamte Zeichenoberfläche dar, die für die BlackBerry®-Geräteanwendung verfügbar ist. Um diesen Bereich zu begrenzen, unterteilen Sie es in XYRect-Objekte. Jedes XYPoint stellt einen Punkt auf dem Bildschirm dar, der sich aus einer X-Koordinate und einer Y-Koordinate zusammensetzt.

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.Graphics`
 - `net.rim.device.api.ui.XYRect`
 - `net.rim.device.api.ui.XYPoint`
2. Erstellen Sie XYPoint-Objekte, um den linken oberen Punkt und den rechten unteren Punkt eines Rechtecks darzustellen. Erstellen Sie mithilfe der XYPoint-Objekte ein XYRect-Objekt, um einen rechteckigen Clipping-Bereich zu erstellen.

```
XYPoint bottomRight = new XYPoint(50, 50);
XYPoint topLeft = new XYPoint(10, 10);
XYRect rectangle = new XYRect(topLeft, bottomRight);
```

3. Rufen Sie `Graphics.pushContext()` auf, um Zeichenaufrufe durchzuführen, die angeben, dass der Bereichsursprung nicht den Zeichen-Offset anpassen soll. Rufen Sie `Graphics.pushContext()` auf, um den rechteckigen Clipping-Bereich mittels Push zum Kontextstapel zu übertragen. Rufen Sie `Graphics.drawRect()` auf, um ein Rechteck zu zeichnen, und rufen Sie `Graphics.fillRect()` auf, um das Rechteck zu füllen. Rufen Sie `Graphics.popContext()` auf, um den aktuellen Kontext vom Kontextstapel loszulösen.

```
graphics.pushContext(rectangle, 0, 0);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
graphics.drawRect(15, 15, 30, 30);
```

```
graphics.pushContext(rectangle, 0, 0);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
graphics.drawRect(15, 15, 30, 30);
```

4. Rufen Sie `pushRegion()` auf, und geben Sie an, dass der Bereichsursprung den Zeichen-Offset anpassen soll. Rufen Sie `Graphics.drawRect()` auf, um ein Rechteck zu zeichnen, und rufen Sie `Graphics.fillRect()` auf, um ein Rechteck zu füllen. Rufen Sie `Graphics.popContext()` auf, um den aktuellen Kontext vom Kontextstapel loszulösen.

```
graphics.pushRegion(rectangle);
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
graphics.popContext();
```

5. Geben Sie den Teil des `Graphics`-Objekts an, der mittels Push zum Stapel übertragen werden soll.
6. Nachdem Sie `pushContext()` (oder `pushRegion()`) aufgerufen haben, stellen Sie den Teil des `Graphics`-Objekts bereit, der invertiert werden soll.

```
graphics.pushContext(rectangle);
graphics.invert(rectangle);
graphics.popContext();
```

7. Rufen Sie `translate()` auf. Das `XYRect` wird von seinem Ursprung (1, 1) in einen Ursprung (20, 20) konvertiert. Nach der Konvertierung reicht der untere Teil des `XYRect`-Objekts über die Grenzen des Grafikkontexts hinaus und schneidet ihn zu.

```
XYRect rectangle = new XYRect(1, 1, 100, 100);
XYPoint newLocation = new XYPoint(20, 20);
rectangle.translate(newLocation);
```

Erstellen eines Bildschirmübergangs

Sie können einen Bildschirmübergang erstellen, um einen visuellen Effekt anzuwenden, der dann angezeigt wird, wenn Ihre Anwendung einen Bildschirm auf einem BlackBerry®-Gerät öffnet oder schließt. Mithilfe der `net.rim.device.api.ui.TransitionContext`-Klasse können Sie die folgenden Arten von Bildschirmübergängen für Ihre Anwendung erstellen.

Übergang	Beschreibung
<code>TRANSITION_FADE</code>	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm durch allmähliches Ein- oder Ausblenden. Dieser Bildschirmübergang erzeugt den visuellen Effekt des Ein- bzw. Ausblendens durch Änderung der Deckkraft des Bildschirms.
<code>TRANSITION_SLIDE</code>	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem dieser auf die Anzeige oder von der Anzeige am Gerät geschoben wird. Sie können Attribute verwenden, um anzugeben, dass sowohl der

Übergang	Beschreibung
	neue als auch der aktuelle Bildschirm geschoben werden, und dadurch ein Effekt erzeugt wird, als würden sich beide Bildschirme bewegen, oder dass sich der neue über den aktuellen Bildschirm schiebt.
TRANSITION_WIPE	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem simuliert wird, dass dieser auf die Anzeige oder von der Anzeige am Gerät gewischt wird.
TRANSITION_ZOOM	Dieser Übergang zeigt einen Bildschirm an oder entfernt einen Bildschirm, indem dieser auf die Anzeige oder von der Anzeige am Gerät gezoomt wird.
TRANSITION_NONE	Es findet kein Übergang statt.

Jede Art von Bildschirmübergang hat Attribute, die Sie verwenden können, um die visuellen Effekte des Bildschirmübergangs anzupassen. Zum Beispiel können Sie einen schiebenden Effekt so anpassen, dass ein Bildschirm vom unteren Bereich des Gerätebildschirms nach oben geschoben wird. Wenn Sie den Bildschirmübergang nicht anpassen, verwendet die Anwendung die Standardattribute. Weitere Informationen zu den Standardattributen finden Sie in der API-Referenz für BlackBerry® Java® Development Environment.

Nachdem Sie einen Bildschirmübergang erstellt haben, müssen Sie ihn innerhalb Ihrer Anwendung registrieren. Dazu rufen Sie `UiEngineInstance.setTransition()` auf und geben den zu entfernenden Ausgangsbildschirm und den anzuzeigenden Eingangsbildschirm sowie die Ereignisse, die den Übergang verursachen, und den anzuzeigenden Übergang an.

Um eine Beispielanwendung herunterzuladen, die die Verwendung von Bildschirmübergängen veranschaulicht, besuchen Sie www.blackberry.com/go/screentransitionssample. Weitere Informationen zu Bildschirmübergängen finden Sie in der API-Referenz für BlackBerry Java Development Environment.

Anmerkung: Die Skin auf dem BlackBerry-Gerät steuert die visuellen Effekte, die angezeigt werden, wenn ein Benutzer eine Anwendung öffnet. Weitere Informationen finden Sie im *BlackBerry Theme Studio-Benutzerhandbuch*.

Codebeispiel: Erstellen eines Bildschirmübergangs

Das folgende Codebeispiel zeigt einen Übergang durch Verschieben sowie einen Übergang durch allmähliches Ausblenden. Wenn der Benutzer die Anwendung öffnet, wird auf dem BlackBerry®-Gerät der erste Bildschirm angezeigt, der eine Schaltfläche enthält. Wenn der Benutzer auf die Schaltfläche klickt, wird von rechts ein zweiter Bildschirm auf den Gerätebildschirm geschoben. Der zweite Bildschirm wird nach zwei Sekunden automatisch ausgeblendet.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
public class ScreenTransitionSample extends UiApplication implements
    FieldChangeListener {
    private Screen _secondaryScreen;
    private Runnable _popRunnable;
    public static void main(String[] args) {
        ScreenTransitionSample theApp = new ScreenTransitionSample ();
        theApp.enterEventDispatcher();
    }
}
```

```

}
public ScreenTransitionSample () {
    _secondaryScreen = new FullScreen();
    _secondaryScreen.setBackground(
        BackgroundFactory.createSolidBackground(Color.LIGHTBLUE) );
    LabelField labelField = new LabelField("The screen closes automatically in
two seconds by using a fade transition");
    _secondaryScreen.add(labelField);
    TransitionContext transition = new
TransitionContext(TransitionContext.TRANSITION_SLIDE);
    transition.setIntAttribute(TransitionContext.ATTR_DURATION, 500);
    transition.setIntAttribute(TransitionContext.ATTR_DIRECTION,
TransitionContext.DIRECTION_RIGHT);
    transition.setIntAttribute(TransitionContext.ATTR_STYLE,
TransitionContext.STYLE_PUSH);
    UiEngineInstance engine = Ui.getUiEngineInstance();
    engine.setTransition(null, _secondaryScreen, UiEngineInstance.TRIGGER_PUSH,
transition);
    transition = new TransitionContext(TransitionContext.TRANSITION_FADE);
    transition.setIntAttribute(TransitionContext.ATTR_DURATION, 500);
    transition.setIntAttribute(TransitionContext.ATTR_KIND,
TransitionContext.KIND_OUT);
    engine.setTransition(_secondaryScreen, null, UiEngineInstance.TRIGGER_POP,
transition);
    MainScreen baseScreen = new MainScreen();
    baseScreen.setTitle("Screen Transition Sample");
    ButtonField buttonField = new ButtonField("View Transition",
ButtonField.CONSUME_CLICK) ;
    buttonField.setChangeListener(this);
    baseScreen.add(buttonField);
    pushScreen(baseScreen);
    _popRunnable = new Runnable() {
        public void run() {
            popScreen(_secondaryScreen);
        }
    };
}
public void fieldChanged(Field field, int context)
{
    pushScreen(_secondaryScreen);
    invokeLater(_popRunnable, 2000, false);
}
}

```

Festlegen des Formats und der Richtung des Bildschirms

In Touchscreen-Anwendungen können Sie sowohl das Format als auch die Richtung des Bildschirms berücksichtigen. Das Format bezieht sich auf das Seitenverhältnis des Bildschirms. Die Richtung bezieht sich auf den Zeichenbereich des Bildschirms.

Format

Der Benutzer eines BlackBerry®-Geräts mit Touchscreen kann das Format des Bildschirms durch Drehen des Geräts ändern. Wenn ein Benutzer das BlackBerry-Gerät so hält, dass sich das BlackBerry-Logo oben befindet, weist der Bildschirm das Hochformat auf. Wenn der Benutzer das Gerät um 90 Grad nach links oder rechts dreht, weist der Bildschirm das Querformat auf.

Sie können die `net.rim.device.api.system.Display`-Klasse verwenden, um die Ausrichtung des Bildschirms abzurufen. Die `Display`-Klasse enthält die Konstanten, die den Formaten entsprechen, die das Gerät mit Touchscreen zum Anzeigen von Informationen verwenden kann. Zum Beispiel entsprechen das Hochformat, Querformat und quadratische Ausrichtungen den `Display.ORIENTATION_PORTRAIT`-, `Display.ORIENTATION_LANDSCAPE`- und `Display.ORIENTATION_SQUARE`-Konstanten.

Um das Format des Bildschirms abzurufen, können Sie die `Display.getOrientation()`-Methode aufrufen. Diese Methode gibt eine Ganzzahl zurück, die einer der Ausrichtungen entspricht, die auf dem BlackBerry-Gerät verwendet werden können.

Damit Ausrichtungsänderungen erkannt werden, können Sie `Screen.sublayout()` überschreiben. Bei dieser Methode rufen Sie `super.sublayout()` auf und suchen nach Änderungen bei den `width`- und `height`-Werten.

Richtung

Ein BlackBerry-Gerät mit Touchscreen kann Informationen auf dem Bildschirm in verschiedenen Richtungen anzeigen. Die Richtung bezieht sich auf das obere Ende des Zeichenbereichs des Bildschirms, das zur Position des BlackBerry-Logos relativ ist. Die Richtung ist "Norden", wenn das obere Ende des Zeichenbereichs die Bildschirmseite ist, die dem BlackBerry-Logo am nächsten liegt, "Westen", wenn sich das obere Ende des Zeichenbereichs links vom BlackBerry-Logo befindet, und "Osten", wenn sich das obere Ende des Zeichenbereichs rechts vom BlackBerry-Logo befindet.

Sie können die `net.rim.device.api.system.Display`-Klasse, die `net.rim.device.api.ui.Ui`-Klasse und die `net.rim.device.api.ui.UiEngineInstance`-Klasse verwenden, um die Richtung zu steuern, die das BlackBerry-Gerät verwendet, um Informationen auf dem Bildschirm anzuzeigen. Die `Display`-Klasse enthält Konstanten, die den Richtungen entsprechen, die das Gerät zum Anzeigen von Informationen verwenden kann. Zum Beispiel entsprechen Norden, Westen und Osten den `Display.DIRECTION_NORTH`-, `Display.DIRECTION_WEST`- und `Display.DIRECTION_EAST`-Konstanten.

Sie können ein Objekt der `net.rim.device.api.ui.Ui`-Klasse erstellen und `Ui.getUiEngineInstance()` aufrufen, um ein Objekt der `UiEngineInstance`-Klasse abzurufen. Durch Aufrufen von `UiEngineInstance.setAcceptableDirections()` mit einer der Konstanten, die Richtungen aus der `Display`-Klasse entsprechen, wird die Richtung festgelegt, die das BlackBerry-Gerät zum Anzeigen von Informationen verwenden kann.

Codebeispiel: Abrufen der Bildschirmausrichtung

```
switch(Display.getOrientation())
{
    case Display.ORIENTATION_LANDSCAPE:
        Dialog.alert("Screen orientation is landscape"); break;
    case Display.ORIENTATION_PORTRAIT:
        Dialog.alert("Screen orientation is portrait"); break;
```

```

case Display.ORIENTATION_SQUARE:
    Dialog.alert("Screen orientation is square"); break;
default:
    Dialog.alert("Screen orientation is not known"); break;
}

```

Codebeispiel: Erzwingen des Hochformats in einer BlackBerry-API-Anwendung

```

// Use code like this before invoking UiApplication.pushScreen()
int direction = Display.DIRECTION_NORTH;
Ui.getUiEngineInstance().setAcceptableDirections(direction);

```

Codebeispiel: Erzwingen des Querformats in einer MIDlet-Anwendung

```

// Use code like this before invoking Display.setCurrent() in the MIDlet constructor
DirectionControl dc =
    (DirectionControl) ((Controllable) Display.getDisplay(this)).
        getControl("net.rim.device.api.lcdui.control.DirectionControl");
int directions = DirectionControl.DIRECTION_EAST | DirectionControl.DIRECTION_WEST;
dc.setAcceptableScreenDirections(directions);

```

Abrufen der Ausrichtung des Touchscreen

1. Importieren Sie die `net.rim.device.api.system.Display`-Klasse.
2. Rufen Sie `net.rim.device.api.system.Display.getOrientation()` auf.

```
int orientation = Display.getOrientation();
```

Schränken Sie die Richtung des Touchscreens ein

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.Ui`
 - `net.rim.device.api.ui.UiEngineInstance`
2. Rufen Sie `net.rim.device.api.ui.Ui.getUiEngineInstance()` auf.

```
UiEngineInstance _ue;
_ue = Ui.getUiEngineInstance();
```

3. Rufen Sie `net.rim.device.api.ui.UiEngineInstance.setAcceptableDirections(byte flags)` auf, und geben Sie das Argument für die Richtung von Screen weiter.

```
_ue.setAcceptableDirections(Display.DIRECTION_WEST);
```

Benachrichtigen lassen, wenn sich die Größe des Zeichenbereichs auf dem Touchscreen ändert

1. Importieren Sie die folgenden Klassen:

- javax.microedition.lcdui.Canvas
 - net.rim.device.api.ui.component.Dialog
2. Erweitern Sie javax.microedition.lcdui.Canvas.
 3. Überschreiben Sie Canvas.sizeChanged(int, int).

```
protected void sizeChanged(int w, int h) {  
    Dialog.alert("The size of the Canvas has changed");  
}
```

Beschleunigungssensor

Ein BlackBerry®-Gerät mit einem Touchscreen enthält einen Beschleunigungssensor, der die Aufgabe hat, die Ausrichtung und die Beschleunigung des BlackBerry-Geräts wahrzunehmen. Wenn ein BlackBerry-Gerätebenutzer das BlackBerry-Gerät bewegt, kann der Beschleunigungssensor die Bewegung im 3D-Bereich entlang der X-, Y- und Z-Achse wahrnehmen. Ein Gerätebenutzer kann die Ausrichtung des Geräts ändern, das die Bildschirmrichtung von einem Bildschirm für eine BlackBerry-Geräteanwendung aus zwischen Hoch- und Querformat ändern kann.

Sie können die Accelerometer APIs im `net.rim.device.api.system`-Paket für die Reaktion auf die Ausrichtung und Beschleunigung des BlackBerry-Geräts verwenden. Zum Beispiel können Sie eine Spielanwendung ändern, um die Richtung und die Geschwindigkeit eines sich bewegenden Objekts auf dem Bildschirm zu ändern, während ein Benutzer das BlackBerry-Gerät mit verschiedenen Geschwindigkeiten bewegt und dreht.

Um eine Beispielanwendung herunterzuladen, die die Verwendung des Beschleunigungssensors veranschaulicht, besuchen Sie www.blackberry.com/go/screentransitionssample. Weitere Informationen zur Beispielanwendung, finden Sie unter www.blackberry.com/go/devguides, im *Accelerometer Sample Application Overview (Überblick über die Beispielanwendung Beschleunigungssensor)*.

Arten von Beschleunigungssensor-Daten

Eine BlackBerry®-Geräteanwendung kann Daten vom Beschleunigungssensor abrufen.

Datentyp	Beschreibung
Ausrichtung	Die Ausrichtung des BlackBerry-Geräts in Bezug auf den Boden.
Beschleunigung	Die Drehbeschleunigung des BlackBerry-Geräts.

Weitere Informationen zu Datentypen des Beschleunigungssensors finden Sie in der API-Referenz für die BlackBerry® Java® Development Environment.

Abrufen von Beschleunigungssensordaten

Der Beschleunigungssensor dient dazu, die Richtung und Geschwindigkeit der Bewegung entlang einer X-, Y- und Z-Achse nachzuverfolgen, wenn ein BlackBerry®-Gerätebenutzer das BlackBerry-Gerät bewegt. Die X-Achse ist zur Breite des BlackBerry-Geräts parallel. Die Y-Achse ist zur Länge des BlackBerry-Geräts parallel. Die Z-Achse ist zur Tiefe (von der Vorder- bis zur Rückseite) des BlackBerry-Geräts parallel. Weitere Informationen zur X-, Y- und Z-Achse des Beschleunigungssensors finden Sie in der Beschreibung der

`net.rim.device.api.system.AccelerometerSensor`-Klasse in der API-Referenz für die BlackBerry® Java® Development Environment.

Sie können einer BlackBerry-Geräteanwendung ermöglichen, auf die Beschleunigung eines BlackBerry-Geräts zu reagieren, das über einen Beschleunigungssensor verfügt. Ein BlackBerry-Gerätebenutzer könnte beispielsweise das BlackBerry-Gerät bewegen, um die Richtung und Geschwindigkeit eines Objekts zu steuern, das sich in einer Spielanwendung durch ein Labyrinth bewegt.

Sie können die Beschleunigungssensor-APIs im `net.rim.device.api.system`-Paket für die Reaktion auf die Beschleunigung des BlackBerry-Geräts verwenden. Sie müssen zuerst bestimmen, ob das BlackBerry-Gerät einen Beschleunigungssensor durch Aufrufen von `net.rim.device.api.system.AccelerometerSensor.isSupported()` unterstützt. Wenn die Methode den Wert `true` zurückgibt, unterstützt das BlackBerry-Gerät einen Beschleunigungssensor.

Mithilfe der `AccelerometerData`-Klasse können Sie die Richtung identifizieren, in der der Benutzer das BlackBerry-Gerät bewegt. Durch Aufrufen von `AccelerometerData.getOrientation()` wird eine der Konstanten der `AccelerometerSensor`-Klasse zurückgegeben, die der Richtung des BlackBerry entsprechen. Wenn `AccelerometerData.getOrientation()` beispielsweise einen Wert gleich `AccelerometerSensor. ORIENTATION_LEFT_UP` zurückgibt, weist die linke Seite des BlackBerry-Geräts nach oben.

Mithilfe der `AccelerometerSensor`-Klasse können Sie Beschleunigungsdaten vom BlackBerry-Gerät abrufen. Durch Aufrufen von `AccelerometerSensor.openRawDataChannel()` wird ein Objekt der `net.rim.device.api.system.AccelerometerSensor.Channel`-Klasse zurückgegeben. Die `AccelerometerSensor.Channel`-Klasse ermöglicht es Ihnen, eine Verbindung zum Beschleunigungssensor herzustellen. Sie können Daten vom Beschleunigungssensor durch Aufrufen von `AccelerometerSensor.Channel.getLastAccelerationData()` abrufen.

Durch die Aufrechterhaltung einer Verbindung zum Beschleunigungssensor wird der Akku des BlackBerry-Geräts beansprucht. Wenn die BlackBerry-Geräteanwendung keine Daten mehr vom Beschleunigungssensor abrufen muss, sollten Sie `AccelerometerSensor.Channel.close()` aufrufen, um die Verbindung zu schließen.

Codebeispiel: Abrufen von Daten vom Beschleunigungssensor

```
short[] xyz = new short[3];
while( running ) {
    channel.getLastAccelerationData(xyz);
}
channel.close();
```

Abrufen der Beschleunigungssensordaten in bestimmten Abständen

Wenn eine BlackBerry®-Geräteanwendung einen Kanal für den Beschleunigungssensor öffnet, während sich die Anwendung im Vordergrund befindet, und die Anwendung sich anschließend im Hintergrund befindet, pausiert der Kanal, und der Beschleunigungssensor wird nicht abgefragt. Wenn eine BlackBerry-Geräteanwendung `AccelerometerSensor.Channel.getLastAccelerationData(short[])` in kurzen Abständen aufruft oder wenn das BlackBerry-Gerät nicht bewegt wird, können durch die Methode doppelte Werte zurückgegeben werden.

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.system.AccelerometerSensor.Channel`;
 - `net.rim.device.api.system.AccelerometerSensor`;
2. Öffnen Sie einen Kanal für den Beschleunigungssensor. Während ein Kanal offen ist, wird von der BlackBerry-Geräteanwendung der Beschleunigungssensor nach Informationen abgefragt.

```
Channel rawDataChannel = AccelerometerSensor.openRawDataChannel  
( Application.getApplication() );
```

- Erstellen Sie ein Array zum Speichern der Beschleunigungssensordaten.

```
short[] xyz = new short[ 3 ];
```

- Erstellen Sie einen Thread.

```
while( running ) {
```

- Rufen Sie `Channel.getLastAccelerationData(short[])` auf, um Daten vom Beschleunigungssensor abzurufen.

```
rawDataChannel.getLastAccelerationData( xyz );
```

- Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
Thread.sleep( 500 );
```

- Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
rawDataChannel.close();
```

Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Vordergrund befindet

- Importieren Sie die folgenden Klassen:

- `net.rim.device.api.system.AccelerometerChannelConfig`
- `net.rim.device.api.system.AccelerometerSensor.Channel`

- Öffnen Sie einen Kanal, um Beschleunigungsdaten vom Beschleunigungssensor abzurufen.

```
Channel channel = AccelerometerSensor.openRawAccelerationChannel  
( Application.getApplication());
```

- Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
short[] xyz = new short[ 3 ];  
while( running ) {  
    channel.getLastAccelerationData( xyz );  
    Thread.sleep( 500 );  
}
```

- Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
channel.close();
```

Abfragen des Beschleunigungssensors, wenn sich die Anwendung im Hintergrund befindet

1. Importieren Sie die folgenden Klassen:

- `net.rim.device.api.system.AccelerometerChannelConfig;`
- `net.rim.device.api.system.AccelerometerSensor.Channel;`

2. Erstellen Sie eine Konfiguration für einen Kanal zum Beschleunigungssensor.

```
AccelerometerChannelConfig channelConfig = new AccelerometerChannelConfig  
( AccelerometerChannelConfig.TYPE_RAW );
```

3. Rufen Sie `AccelerometerChannelConfig.setBackgroundMode( Boolean )` auf, um den Support für eine Anwendung anzugeben, die sich im Hintergrund befindet.

```
channelConfig.setBackgroundMode( true );
```

4. Rufen Sie `AccelerometerSensor.openChannel()` auf, um einen Kanal, der sich im Hintergrund befindet, zum Beschleunigungssensor zu öffnen.

```
Channel channel = AccelerometerSensor.openChannel( Application.getApplication(),  
channelConfig );
```

5. Rufen Sie `Thread.sleep()` auf, um den Abstand zwischen Abfragen zum Beschleunigungssensor in Millisekunden anzugeben.

```
short[] xyz = new short[ 3 ];  
while( running ) {  
    channel.getLastAccelerationData( xyz );  
    Thread.sleep( 500 );  
}
```

6. Rufen Sie `Channel.close()` auf, um den Kanal zum Beschleunigungssensor zu schließen.

```
channel.close();
```

Speichern der ausgelesenen Beschleunigungssensorwerte in einem Puffer

1. Importieren Sie die folgenden Klassen:

- `net.rim.device.api.system.AccelerometerChannelConfig;`
- `net.rim.device.api.system.AccelerometerSensor.Channel;`

2. Erstellen Sie eine Konfiguration für einen Kanal zum Beschleunigungssensor.

```
AccelerometerChannelConfig channelConfig = new AccelerometerChannelConfig  
( AccelerometerChannelConfig.TYPE_RAW );
```

3. Rufen Sie `AccelerometerChannelConfig.setSamplesCount()` auf, um die Anzahl der im Puffer zu speichernden ausgelesenen Beschleunigungswerte anzugeben. Jedes Element im Puffer beinhaltet die ausgelesenen Beschleunigungswerte für die X-, Y- und Z-Achse sowie Daten über den Zeitpunkt des Auslesens.

```
channelConfig.setSamplesCount( 500 );
```

4. Rufen Sie `AccelerometerSensor.openChannel()` auf, um einen Kanal zum Beschleunigungssensor zu öffnen.

```
Channel bufferedChannel = AccelerometerSensor.openChannel  
( Application.getAppApplication(), channelConfig );
```

Abrufen von Beschleunigungssensordaten aus einem Puffer

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.system.AccelerometerData;`
 - `net.rim.device.api.system.AccelerometerSensor.Channel;`
2. Fragen Sie Beschleunigungssensordaten aus dem Puffer ab.

```
AccelerometerData accData = bufferedChannel.getAccelerometerData();
```

3. Rufen Sie `AccelerometerData.getNewBatchLength()` auf, um die Anzahl der Werte zu erhalten, die seit der letzten Abfrage abgerufen wurden.

```
int newBatchSize = accData.getNewBatchLength();
```

4. Rufen Sie `AccelerometerData.getXAccHistory()`, `AccelerometerData.getYAccHistory()` und `AccelerometerData.getZAccHistory()` auf, um Beschleunigungssensordaten vom Puffer für jede Achse abzurufen.

```
short[] xAccel = accData.getXAccHistory();  
short[] yAccel = accData.getYAccHistory();  
short[] zAccel = accData.getZAccHistory();
```

Ermitteln des Zeitpunkts des Abrufens eines Werts vom Beschleunigungssensor

1. Importieren Sie die `net.rim.device.api.system.AccelerometerData`-Klasse.
2. Fragen Sie Beschleunigungssensordaten aus dem Puffer ab.

```
AccelerometerData accData;  
accData = bufferedChannel.getAccelerometerData();
```

3. Rufen Sie `AccelerometerData.getSampleTsHistory()` auf.

```
long[] queryTimestamps = accData.getSampleTsHistory();
```

Ereignisse

5

Reagieren auf Navigationsereignisse

Sie können die Screen-Navigationsmethoden verwenden, um eine BlackBerry®-Geräteanwendung zu erstellen. Wenn Ihre vorhandene BlackBerry-Geräteanwendung die TrackwheelListener-Schnittstelle implementiert, aktualisieren Sie Ihre BlackBerry-Geräteanwendung, um die Screen-Navigationsmethoden zu verwenden.

1. Importieren Sie die `net.rim.device.api.ui.Screen`-Klasse.
2. Verwalten Sie Navigationsereignisse durch Erweitern der `net.rim.device.api.ui.Screen`-Klasse (oder einer ihrer Unterklassen), und überschreiben Sie die folgenden Navigationsmethoden:

```
navigationClick(int status, int time)
navigationUnclick(int status, int time)
navigationMovement(int dx, int dy, int status, int time)
```

Bestimmen der Art der Eingabemethode

1. Importieren Sie eine oder mehrere der folgenden Klassen:
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.Field`
2. Importieren Sie die `net.rim.device.api.system.KeypadListener`-Schnittstelle.
3. Implementieren Sie die `net.rim.device.api.system.KeypadListener`-Schnittstelle.
4. Erweitern Sie die `Screen`-Klasse, die `Field`-Klasse oder beide.
5. In Ihrer Implementierung der Methoden `navigationClick`, `navigationUnclick` oder `navigationMovement` führen Sie einen bitweisen AND-Vorgang auf dem `status`-Parameter aus, um mehr Informationen zum Ereignis zu erzeugen. Implementieren Sie die `navigationClick(int status, int time)`-Methode, um zu bestimmen, ob das Trackwheel oder ein Vier-Wege-Navigationseingabegerät (zum Beispiel ein Trackball) ein Ereignis auslöst.

```
public boolean navigationClick(int status, int time) {
    if ((status & KeypadListener.STATUS_TRACKWHEEL) ==
        KeypadListener.STATUS_TRACKWHEEL) {
        //Input came from the trackwheel
    } else if ((status & KeypadListener.STATUS_FOUR_WAY) ==
        KeypadListener.STATUS_FOUR_WAY) {
        //Input came from a four way navigation input device
    }
    return super.navigationClick(status, time);
}
```

Weitere Informationen zur Statusmodifizierung für die `net.rim.device.api.system.KeypadListener`-Klasse finden Sie in der API-Referenz für die BlackBerry® Java® Development Environment.

Modelle für Touchscreen-Interaktionen

BlackBerry® Geräte mit Touchscreen und Trackpad verwenden ein Modell mit ganz leichten Klick-Interaktionen, bei dem der BlackBerry-Gerätebenutzer den Bildschirm antippt oder das Trackpad verwendet, um eine Aktion auszuführen. Dieses Modell weicht von vorherigen BlackBerry-Geräten mit einem SurePress™-Touchscreen ab, bei denen der Benutzer auf den Bildschirm klickt (drückt), um eine Aktion auszuführen.

Bei beiden Gerätearten tritt ein TouchEvent . DOWN-Ereignis auf, wenn der Benutzer den Bildschirm berührt.

Bei Geräten mit einem SurePress-Touchscreen tritt ein TouchEvent . CLICK-Ereignis auf, wenn der Benutzer Druck anwendet und auf den Bildschirm klickt. Wenn der Druck verringert wird, tritt ein TouchEvent . UNCLICK-Ereignis auf. In der Regel treten bei einem TouchEvent . UNCLICK-Ereignis Aktionen (wie z. B. das auf einer Schaltfläche basierende Aufrufen einer Aktion) ein. Wenn der Benutzer den Bildschirm komplett loslässt, tritt ein TouchEvent . UP-Ereignis auf.

Auf Geräten mit einem Touchscreen, der leichte Klick-Interaktionen unterstützt, gibt es kein physisches Anklicken und Deaktivieren (durch erneutes Anklicken). Stattdessen treten die Aktionen dann auf, wenn der Benutzer den Bildschirm antippt. Antippen ist ein TouchEvent . DOWN-Ereignis, auf das schnell ein TouchEvent . UP-Ereignis folgt. Diese Kombination von Berührungseignissen wird als Geste bezeichnet, weswegen auch ein TouchGesture auftritt. Bei einem Antippen tritt ein TouchGesture . TAP-Ereignis (zusammen mit dem TouchEvent . DOWN- und TouchEvent . UP-Ereignis) auf.

Für eine größere Kompatibilität zwischen den beiden Modellarten für Touchscreen-Interaktionen, treten auf Geräten, die leichte Klick-Interaktionen unterstützen, die beiden Ereignisse TouchEvent . CLICK und TouchEvent . UNCLICK nach dem TouchGesture . TAP-Ereignis auf, sodass Sie beim TouchEvent . UNCLICK-Ereignis eine Aktion auf allen Touchscreen-Geräten aufrufen können, und das unabhängig davon, ob das Gerät über einen SurePress-Touchscreen verfügt oder nicht.

Benutzeraktion	Geräte, die leichte Klick-Interaktionen unterstützen	Geräte mit einem SurePress-Touchscreen
Berühren des Bildschirms.	TouchEvent . DOWN	TouchEvent . DOWN
Berühren eines Element und mit dem Finger halten.	TouchGesture . HOVER	TouchGesture . HOVER
Klicken auf den Bildschirm.	—	TouchEvent . CLICK
Klicken auf den Bildschirm und halten.	—	TouchEvent . CLICK_REPEAT
Loslassen des Bildschirms.	—	TouchEvent . UNCLICK
Entfernen eines Fingers vom Bildschirm.	TouchEvent . UP	TouchEvent . UP

Benutzeraktion	Geräte, die leichte Klick-Interaktionen unterstützen	Geräte mit einem SurePress-Touchscreen
Antippen des Bildschirms.	<code>TouchEvent.DOWN</code>	<code>TouchEvent.DOWN</code>
	<code>TouchGesture.TAP</code>	<code>TouchGesture.TAP</code>
	<code>TouchEvent.CLICK</code>	<code>TouchEvent.UP</code>
	<code>TouchEvent.UNCLICK</code>	
	<code>TouchEvent.UP</code>	

Sie können bestimmen, ob ein Gerät mit einem Touchscreen leichte Klick-Interaktionen unterstützt, indem Sie `DeviceCapability.isTouchClickSupported()` aufrufen. Wenn das Gerät leichte Klick-Interaktionen unterstützt, gibt die Methode `false` zurück. Wenn das Gerät über einen SurePress-Touchscreen verfügt, gibt die Methode `true` zurück.

Reagieren auf Touchscreen-Ereignisse

BlackBerry®-Gerätebenutzer können auf einem Gerät mit Touchscreen die gleichen Aktionen wie auf einem BlackBerry-Gerät mit Trackball ausführen. Beispielsweise können BlackBerry-Gerätebenutzer den Touchscreen verwenden, um ein Menü zu öffnen, durch eine Liste von Optionen zu scrollen und eine Option auszuwählen.

Wenn Sie mithilfe einer älteren Version als Version 4.7 der BlackBerry® Java® Development Environment eine BlackBerry-Geräteanwendung erstellen, können Sie die JAD-Datei der Anwendung ändern, damit die Anwendung reagieren kann, wenn ein BlackBerry-Gerätebenutzer den Touchscreen berührt. Legen Sie in der JAD-Datei für die Option `RIM-TouchCompatibilityMode false` fest.

Wenn Sie BlackBerry JDE Version 4.7 oder höher verwenden, um eine BlackBerry-Geräteanwendung zu erstellen, können Sie die Anwendung aktivieren, damit sie die Aktion identifizieren kann, die der BlackBerry-Gerätebenutzer auf dem Touchscreen durch Erweitern der `net.rim.device.api.ui.Screen`-Klasse, der `net.rim.device.api.ui.Field`-Klasse oder einer der Unterklassen der `Field`-Klasse und das Überschreiben der `touchEvent()`-Methode ausführt. Innerhalb der `touchEvent()`-Methode vergleichen Sie den Wert, den `TouchEvent.getEvent()` an die Konstanten von der `net.rim.device.api.ui.TouchEvent`-Klasse und der `net.rim.device.api.ui.TouchGesture`-Klasse aus zurückgibt.

Die `TouchEvent`-Klasse enthält die Konstanten, die die verschiedenen Aktionen darstellen, die ein Benutzer auf dem Touchscreen ausführen kann. Zum Beispiel entsprechen Klick-, Berührungs- und Bewegungsaktionen den `TouchEvent.CLICK`-, `TouchEvent.DOWN`- und `TouchEvent.MOVE`-Konstanten.

Die `TouchGesture`-Klasse enthält die Konstanten, die die verschiedenen Gesten darstellen, die ein Benutzer auf einem Touchscreen ausführen kann. Zum Beispiel entsprechen Tipp- und Fingerstrich-Gesten den `TouchGesture.TAP`- und `TouchGesture.SWIPE`-Konstanten.

Codebeispiel: Identifizieren der Aktion, die ein Benutzer auf dem Touchscreen ausführt

Im folgenden Codebeispiel wird eine `switch`-Anweisung zur Identifizierung der Aktion verwendet.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.CLICK:
            Dialog.alert("A click action occurred");
            return true;
        case TouchEvent.DOWN:
            Dialog.alert("A down action occurred");
            return true;
        case TouchEvent.MOVE:
            Dialog.alert("A move action occurred");
            return true;
    }
    return false;
}
```

Antworten auf Systemereignisse, während der Benutzer den Bildschirm berührt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchEvent.getEvent()` zurückgibt, gleich `TouchEvent.CANCEL` ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.CANCEL:
            Dialog.alert("System event occurred while processing touch events");
            return true;
    }
    return false;
}
```

Reagieren auf einen Benutzer, der einen Finger schnell nach oben über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`

- `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchGesture.getSwipeDirection()` zurückgibt, gleich `TouchGesture.SWIPE_NORTH` ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
                case TouchGesture.SWIPE:
                    if(gesture.getSwipeDirection() == TouchGesture.SWIPE_NORTH) {
                        Dialog.alert("Upward swipe occurred");
                        return true;
                    }
            }
            return false;
    }
}
```

Reagieren auf einen Benutzer, der einen Finger schnell nach unten über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.

4. Prüfen Sie, ob der Wert, den `TouchEvent.getSwipeDirection()` zurückgibt, gleich `TouchEvent.SWIPE_SOUTH` ist.

```
protected boolean onTouchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
                case TouchGesture.SWIPE:
                    if(gesture.getSwipeDirection() == TouchGesture.SWIPE_SOUTH) {
                        Dialog.alert("Downward swipe occurred");
                        return true;
                    }
            }
            return false;
    }
}
```

Reagieren auf einen Benutzer, der einen Finger schnell nach rechts über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `onTouchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchEvent.getSwipeDirection()` zurückgibt, gleich `TouchEvent.SWIPE_EAST` ist.

```
protected boolean onTouchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
                case TouchGesture.SWIPE:
                    if(gesture.getSwipeDirection() == TouchGesture.SWIPE_EAST) {
                        Dialog.alert("Eastward swipe occurred");
                        return true;
                    }
            }
    }
}
```

```
        return false;
    }
}
```

Reagieren auf einen Benutzer, der schnell einen Finger nach rechts über den Bildschirm bewegt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchGesture.getSwipeDirection()` zurückgibt, gleich `TouchGesture.SWIPE_WEST` ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
                case TouchGesture.SWIPE:
                    if(gesture.getSwipeDirection() == TouchGesture.SWIPE_WEST) {
                        Dialog.alert("Westward swipe occurred");
                        return true;
                    }
            }
            return false;
    }
}
```

Reagieren auf einen Benutzer, der auf den Bildschirm klickt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`

2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchEvent.getEvent()` zurückgibt, gleich `TouchEvent.CLICK` ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.CLICK:
            Dialog.alert("CLICK occurred");
            return true;
    }
    return false;
}
```

Reagieren auf einen Benutzer, der den Bildschirm zweimal schnell berührt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Überprüfen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode, ob ein `TouchGesture.TAP`-Ereignis vorliegt und ob `TouchGesture.getTapCount` 2 zurückgibt.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent()) {
                case TouchGesture.TAP:
                    if(gesture.getTapCount() == 2) {
                        Dialog.alert("Double tap occurred");
                        return true;
                    }
            }
    }
    return false;
}
```

Reagieren auf einen Benutzer, der ein Element auf dem Bildschirm berührt und zieht

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchEvent.getEvent()` zurückgibt, gleich `TouchEvent.MOVE` ist.

```
protected boolean touchEvent(TouchEvent message) {  
    switch(message.getEvent()) {  
        case TouchEvent.MOVE:  
            Dialog.alert("Move event occurred");  
            return true;  
        }  
    return false;  
}
```

Reagieren auf einen Benutzer, der den Bildschirm leicht berührt

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.TouchEvent`
 - `net.rim.device.api.ui.TouchGesture`
 - `net.rim.device.api.ui.Field`
 - `net.rim.device.api.ui.Manager`
 - `net.rim.device.api.ui.Screen`
 - `net.rim.device.api.ui.component.Dialog`
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Rufen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode `TouchEvent.getEvent()` auf.
4. Prüfen Sie, ob der Wert, den `TouchEvent.getEvent()` zurückgibt, gleich `TouchEvent.DOWN` ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.DOWN:
            Dialog.alert("Touch occurred");
            return true;
        }
    return false;
}
```

Reagieren auf eine Bildlaufaktion

1. Importieren Sie die folgenden Klassen:
 - net.rim.device.api.ui.TouchEvent
 - net.rim.device.api.ui.TouchGesture
 - net.rim.device.api.ui.Field
 - net.rim.device.api.ui.Manager
 - net.rim.device.api.ui.Screen
2. Erstellen Sie eine Klasse, die die Manager-Klasse erweitert.

```
public class newManager extends Manager {
```

3. Prüfen Sie in Ihrer Implementierung der touchEvent(TouchEvent message)-Methode, ob der Wert, den TouchEvent.getEvent() zurückgibt, gleich TouchEvent.MOVE ist.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.MOVE:
            return true;
        }
    return false;
}
```

Reagieren auf einen Benutzer, der den Bildschirm an zwei Stellen gleichzeitig berührt

1. Importieren Sie die folgenden Klassen:
 - net.rim.device.api.ui.TouchEvent
 - net.rim.device.api.ui.Field
 - net.rim.device.api.ui.Manager
 - net.rim.device.api.ui.Screen
 - net.rim.device.api.ui.component.Dialog
2. Erstellen Sie eine Klasse, die die Manager-Klasse, die Screen-Klasse, die Field-Klasse oder eine der Field-Unterklassen erweitert.

```
public class newButtonField extends ButtonField {
```

3. Prüfen Sie in Ihrer Implementierung der `touchEvent(TouchEvent message)`-Methode, ob die folgenden Methodenaufrufe Werte zurückgeben, die größer Null sind: `TouchEvent.getX(1)`, `TouchEvent.getY(1)`, `TouchEvent.getX(2)`, `TouchEvent.getY(2)`.

```
protected boolean touchEvent(TouchEvent message) {
    switch(message.getEvent()) {
        case TouchEvent.MOVE:
            if(message.getX(1) > 0 && message.getY(1) > 0) {
                Dialog.alert("First finger touched/moved");
            }
            if(message.getX(2) > 0 && message.getY(2) > 0) {
                Dialog.alert("Second finger touched/moved");
            }
            return true;
        }
    return false;
}
```

Pinch-Gesten

Pinch-Gesten werden auf BlackBerry®-Geräten mit Touchscreen unterstützt. Pinch-Gesten vergrößern und verkleinern in der Regel Objekte auf dem Bildschirm.

Ein Pinch beginnt, wenn zwei Berührungspunkte auf dem Touchscreen erkannt werden. Der Kernpunkt eines Pinch wird als Mittelpunkt zwischen den beiden anfänglichen Berührungspunkten definiert.

- Wenn zwei Berührungspunkte erkannt werden, tritt ein `TouchEvent.PINCH_BEGIN`-Ereignis auf. Die Koordinatenpunkte des Kernpunkts der Pinch-Geste werden als `TouchEvent.getX(1)` und `TouchEvent.getY(1)` gespeichert. Sie können auf den anfänglichen Vergrößerungsfaktor zugreifen, indem Sie `TouchEvent.getPinchMagnitude()` aufrufen.
- Wenn der Benutzer einen oder beide Berührungspunkte verschiebt, tritt eine Reihe von `TouchEvent.PINCH_UPDATE`-Ereignissen auf. Sie können auf die Vergrößerungsfaktoren während der Pinch-Ausführung zugreifen, indem Sie `TouchEvent.getPinchMagnitude()` aufrufen.
- Wenn der Benutzer die Geste abschließt, tritt ein `TouchEvent.PINCH_END`-Ereignis auf. Sie können auf den endgültigen Vergrößerungsfaktor zugreifen, indem Sie `TouchEvent.getPinchMagnitude()` aufrufen.

Codebeispiel: Abrufen von Informationen zu einer Pinch-Geste

Dieses Beispiel veranschaulicht, wie Pinch-Gesten in der `touchEvent()`-Methode eines Bildschirms verarbeitet werden.

```
protected boolean touchEvent(TouchEvent message)
{
    switch(message.getEvent())
    {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent())
            {
                case TouchGesture.PINCH_END:
```

```

        Dialog.alert("Focal point: " + message.getX(1)
+                  ", " + message.getY(1)
+                  "\nFinal zoom value: " + gesture.getPinchMagnitude());
        return true;
    }
}
return false;
}

```

Aktivieren von Pinch-Gesten

Standardmäßig sind Pinch-Gesten auf dem Bildschirm einer BlackBerry®-Geräteanwendung nicht aktiviert. Sie müssen die Bildeigenschaft festlegen, um die Generierung von Pinch-Gesten auf einem Bildschirm zu aktivieren.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.input.*;

```

2. Erstellen Sie in Ihrem Bildschirmobjekt ein `InputSettings`-Objekt, um es mit Touchscreen-Eigenschaften aufzufüllen.

```

InputSettings ts = TouchscreenSettings.createEmptySet();

```

3. Setzen Sie die Eigenschaft `TouchscreenSettings.DETECT_PINCH` auf 1.

```

ts.set(TouchscreenSettings.DETECT_PINCH, 1);

```

4. Rufen Sie `Screen.addInputSettings()` auf, um dem Bildschirm die Eingabeeinstellungen hinzuzufügen.

```

addInputSettings(ts);

```

Sie können den Wert der `TouchscreenSettings.DETECT_PINCH`-Eigenschaft jederzeit und nicht nur bei der Erstellung des Bildschirms ändern.

Codebeispiel

Dieses Beispiel veranschaulicht, wie Sie Pinch-Gesten im Konstruktor eines Bildschirms aktivieren.

```

public MyScreen()
{
    setTitle("Sample Screen");
    InputSettings ts = TouchscreenSettings.createEmptySet();
    ts.set(TouchscreenSettings.DETECT_PINCH, 1);
    addInputSettings(ts);
    ...
}

```

Fingerstrich-Gesten mit Trackpads

Ähnlich zu BlackBerry®-Geräten mit Touchscreen, unterstützen Geräte mit Trackpad Fingerstrich-Gesten, die BlackBerry-Gerätebenutzer mit dem Trackpad machen.

Wenn der Benutzer den Finger über das Trackpad führt, wird ein `TouchEvent`-Objekt mit dem Ereignistyp `TouchEvent.NAVIGATION_SWIPE` erstellt. Sie können mit den folgenden Methoden Informationen zu einem Trackpad-Fingerstrich abrufen:

- `TouchEvent.getSwipeAngle()`
- `TouchEvent.getSwipeDirection()`
- `TouchEvent.getSwipeContentAngle()`
- `TouchEvent.getSwipeContentDirection()`
- `TouchEvent.getSwipeMagnitude()`

Codebeispiel: Abrufen von Informationen zu einem Trackpad-Fingerstrich

Dieses Beispiel veranschaulicht, wie Trackpad-Fingerstriche in der `TouchEvent()`-Methode eines Bildschirms verarbeitet werden.

```
protected boolean onTouchEvent(TouchEvent message)
{
    switch(message.getEvent())
    {
        case TouchEvent.GESTURE:
            TouchGesture gesture = message.getGesture();
            switch(gesture.getEvent())
            {
                case TouchGesture.NAVIGATION_SWIPE:
                    Dialog.alert("Swipe direction: " + gesture.getSwipeDirection()
                        + ", "
                        + "\nMagnitude: " + gesture.getSwipeMagnitude());
                    return true;
            }
        }
    }
    return false;
}
```

Aktivieren von Fingerstrich-Gesten, die Benutzer auf dem Trackpad machen

Standardmäßig sind Fingerstrich-Gesten, die BlackBerry®-Gerätebenutzer auf dem Trackpad machen, nicht auf dem Bildschirm aktiviert. Sie müssen die `Navigation`-Eigenschaft so festlegen, dass auf dem Bildschirm das Erzeugen von Fingerstrich-Gesten über das Trackpad aktiviert ist.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.input.*;
```

2. Erstellen Sie in Ihrem Bildschirmobjekt ein `InputSettings`-Objekt, um es mit Navigationseigenschaften aufzufüllen.

```
InputSettings nd = NavigationDeviceSettings.createEmptySet();
```

3. Setzen Sie die Eigenschaft `NavigationDeviceSettings.DETECT_SWIPE` auf 1.

```
nd.set(NavigationDeviceSettings.DETECT_SWIPE, 1);
```

4. Rufen Sie `Screen.addInputSettings()` auf, um dem Bildschirm die Eingabeeinstellungen hinzuzufügen.

```
addInputSettings(nd);
```

Sie können den Wert der `NavigationDeviceSettings.DETECT_SWIPE`-Eigenschaft jederzeit und nicht nur bei der Erstellung des Bildschirms ändern.

Codebeispiel

Dieses Beispiel veranschaulicht, wie Sie im Konstruktor eines Bildschirms Fingerstrich-Gesten über das Trackpad aktivieren.

```
public MyScreen()
{
    setTitle("Sample Screen");
    InputSettings nd = NavigationDeviceSettings.createEmptySet();
    nd.set(NavigationDeviceSettings.DETECT_SWIPE, 1);
    addInputSettings(nd);
    ...
}
```

Ereignisinjektion

Mithilfe der `EventInjector`-Klasse und ihrer inneren Klassen können Sie programmgesteuert Benutzeroberflächenereignisse generieren. Auf BlackBerry®-Geräten, die BlackBerry® Device Software Version 5.0 oder höher ausführen und Touchscreens besitzen, können Sie Berührungseignisse, wie z. B. Fingerstriche und Antippen, injizieren. Mithilfe einer der inneren `EventInjector`-Klassen können Sie ein Ereignis modellieren, und mit der `invokeEvent()`-Methode können Sie das Ereignis injizieren. Das Ereignis wird an die Anwendung gesendet, die derzeit ausgewählt ist und auf die Eingabe wartet.

Sie können die Ereignisinjektion verwenden, um Tests zu automatisieren. Sie können die Ereignisinjektion auch verwenden, um Peripheriegeräten die Möglichkeit zu geben, mit einem BlackBerry-Gerät zu interagieren. Sie können sie auch in barrierefreien Anwendungen wie z. B. Sprache-in-Text-Konvertieren verwenden. Für eine Beispielanwendung, die die Ereignisinjektion demonstriert, besuchen Sie www.blackberry.com/go/toucheventinjectorsampleapp, um die Beispielanwendung des Berührungseignisinjektors herunterzuladen. Weitere Informationen zur Beispielanwendung finden Sie unter www.blackberry.com/go/docs/developers im *Touch Event Injector Sample Application Overview (Überblick über die Beispielanwendung des Berührungseignisinjektors)*.

Command Framework API

Sie können das Command Framework API im `net.rim.device.api.command`-Paket verwenden, um die Funktionalität zu definieren, die Sie an verschiedenen Positionen in Ihren eigenen oder in anderen Anwendungen auf dem BlackBerry®-Gerät verwenden können.

Das Command Framework API enthält Funktionen für die Befehlsverarbeitung sowie Befehlsmetadaten und Befehle.

Element	Beschreibung
Befehlsverarbeitung	Sie verwenden die <code>CommandHandler</code> -Klasse, um die Funktionalität zu definieren, die Sie in Ihrer Anwendung oder in anderen Anwendungen auf dem Gerät bereitstellen möchten. Sie erstellen eine Klasse, die die abstrakte <code>CommandHandler</code> -Klasse erweitert und definieren die Funktionalität in der Klassenmethode <code>execute()</code> . Diese Funktionalität wird Befehl genannt.
Befehlsmetadaten	Sie verwenden die <code>CommandMetadata</code>-Klasse, um Metadaten zu definieren, die einen Befehl beschreiben. Die Metadaten der jeweiligen Befehle werden in einem <code>CommandMetadata</code>-Objekt eingebunden. In den Metadaten eines Befehl muss als einziger Inhalt die Befehls-ID enthalten sein, die bei der Erstellung des <code>CommandMetadata</code>-Objekts bereitgestellt und im <code>COMMAND_ID</code>-Feld des Objekts gespeichert wird. <code>CommandMetadata</code> stellt weitere Felder bereit, die Sie zur Beschreibung des Befehls verwenden können. Dazu gehört z. B. das Feld <code>RESOURCE_BUNDLE_NAME</code>, zu dem Sie den Namen des vom Befehl verwendeten Ressourcenpakets zuweisen können. Sie können außerdem Metadatenelemente definieren, die spezifisch für einen bestimmten Befehl sind.
Befehl	Sie verwenden die <code>Command</code> -Klasse, um einen Befehl auszuführen. Sie können sich eine <code>Command</code> -Instanz als ein Proxy zu einer Klasseninstanz vorstellen, die <code>CommandHandler</code> erweitert. Beim Aufruf von <code>Command.execute()</code> wird dieser an die zugewiesene <code>CommandHandler</code> -Instanz delegiert. Zur Ausführung erfolgt dabei eine Übergabe des aktuellen Kontext sowie eine transparente Übergabe einer schreibgeschützten Version der zugewiesenen Befehlsmetadaten.

Das Command Framework API stellt zwei Registries bereit, die Sie zum Speichern der Befehle verwenden können:

- Sie können die lokale Registry zum Speichern von Befehlen nutzen, die Sie in Ihrer Anwendung mithilfe der `LocalCommandRegistrarConnection`-Klasse verwenden.
- Sie können die globale Registry zum Speichern von Befehlen nutzen, die Sie in anderen Anwendungen auf dem Gerät mithilfe der `RemoteCommandRegistrarConnection`-Klasse verwenden.

Die Command Framework Demo und Command Framework Demo Remote App-Beispielanwendungen sind im BlackBerry® Java® SDK enthalten. Diese Beispielanwendungen veranschaulichen, wie Befehle in einer einzelnen Anwendung definiert und verwendet werden und wie Befehle in einer Anwendung definiert und dann in anderen Anwendungen verwendet werden.

Verwenden eines Befehls mit einer Benutzeroberflächenkomponente

Sie können einen Befehl zur Verwendung mit einer Benutzeroberflächenkomponente in Ihrer BlackBerry®-Geräteanwendung definieren. Wenn Sie diese Methode verwenden, müssen Sie keine Metadaten für den Befehl definieren. Zu den zentralen Benutzeroberflächenkomponenten für BlackBerry-Geräteanwendungen, die Befehle unterstützen, gehören Menüeinträge, Schaltflächen, Popup-Menüeinträge und Werkzeugleisten.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.command.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie eine Befehlsverarbeitung, indem Sie eine Klasse erstellen, die die abstrakte `CommandHandler`-Klasse erweitert. Definieren Sie in `execute()` die Funktionalität, die Sie mit der Benutzeroberflächenkomponente verknüpfen möchten. Um festzulegen, ob ein Befehl für ein bestimmtes Kontextobjekt ausführbar ist, können Sie eine `canExecute()`-Methode in Ihrer Befehlsverarbeitung implementieren. Wenn Sie `canExecute()` nicht implementieren, ist Ihr Befehl immer ausführbar.

```
// this command is always executable
class DialogCommandHandler extends CommandHandler
{
    public void execute(ReadOnlyCommandMetadata metadata, Object context)
    {
        Dialog.alert("Executing command for " + context.toString());
    }
}
```

3. Erstellen Sie die Benutzeroberflächenkomponente.

```
MenuItem myItem = new MenuItem(new StringProvider("My Menu Item"),
    0x230000, 0);
```

4. Bei einem Menüeintrag oder einer Schaltfläche können Sie optional den Kontext für den Befehl mithilfe der `setCommandContext()`-Methode der Benutzeroberflächenkomponente festlegen. Für einige Befehle ist möglicherweise ein Kontext erforderlich, um zu bestimmen, was der Befehl machen soll. Wenn Sie keinen Kontext festlegen, wird das Menüeintrags- oder Schaltflächenobjekt zum Kontext.

```
myItem.setCommandContext(new Object()
{
    public String toString()
    {
        return "My MenuItem Object";
    }
});
```

5. Rufen Sie `setCommand()` auf, um den Befehl für die Benutzeroberflächenkomponente festzulegen und die Befehlsverarbeitung als Argument der Methode bereitzustellen.

```
myItem.setCommand(new Command(new DialogCommandHandler()));
```

6. Fügen Sie Ihrem Bildschirm die Komponente hinzu.

```
addItem(myItem);
```

Der Befehl wird ausgeführt, wenn ein BlackBerry-Gerätebenutzer eine Aktion mit der Benutzeroberflächenkomponente ausführt (zum Beispiel wenn der Benutzer auf einen Menüeintrag klickt).

Codebeispiele

Beispiele dieser Methode zum Definieren und Verwenden von Befehlen finden Sie in der API-Referenz zu [MenuItem](#) und [ButtonField](#).

Verwenden eines Befehl in mindestens einer Anwendung

Sie können einen Befehl in einer lokalen Registry – zur Verwendung in der eigenen Anwendung – oder in einer globalen Registry – zur Verwendung in einer beliebigen Anwendung – auf dem BlackBerry®-Gerät speichern.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.command.*;
import net.rim.device.api.command.registrar.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie eine Befehlsverarbeitung, indem Sie eine Klasse erstellen, die die abstrakte `CommandHandler`-Klasse erweitert. Definieren Sie in `execute()` die Funktionalität, die Sie bereitstellen möchten. Um festzulegen, ob ein Befehl für ein bestimmtes Kontextobjekt ausführbar ist, können Sie `canExecute()` optional in Ihrer Befehlsverarbeitung implementieren. Wenn Sie `canExecute()` nicht implementieren, ist Ihr Befehl immer ausführbar.

```
private static class ViewContactCommand extends CommandHandler
{
    public void execute(ReadOnlyCommandMetadata metadata, Object context)
    {
        // Invoke the Contacts application to view a contact here
    }
    public boolean canExecute(ReadOnlyCommandMetadata metadata, Object context)
    {
        // Return true if the command can be executed
        // with the passed-in context object.
    }
}
```

3. Definieren Sie die Metadaten des Befehls.

```
String id = "com.example.contacts.view.emailaddr";
CommandMetadata metadata = new CommandMeta(id);
```

4. Wenn Anwendungen, die den Befehl verwenden, Instanzen vom Befehl erstellen müssen, speichern Sie den Klassennamen des Befehls in den Metadaten.

```
metadata.setClassname(ViewContactCommand.class.getName());
```

Anmerkung: Anwendungen können Instanzen von Drittanbieterbefehlen nur dann dynamisch erstellen, wenn die Befehle in der lokalen Registry gespeichert sind.

5. Definieren Sie optional Befehls- und Kontextkategorien für den Befehl. Anwendungen, die den Befehl verwenden, nutzen die Kategorien, um Befehle in einer Registry zu finden und um zu bestimmen, ob ein Befehl ausführbar ist.

Kategorietyp	Beschreibung
Befehl	Legt eine Kategorisierungsart für den Befehl fest. Anwendungen, die Befehle verwenden, können eine Registry nach Befehlen in festgelegten Kategorien abfragen.
Kontext	Legt den für Ihren Befehl geeigneten Ausführungskontext fest. Anhand dieser Kategorie können Sie den Anwendungen, die den Befehl verwenden, mitteilen, welche Kontextobjektarten an <code>CommandHandler.Execute()</code> übergeben werden können.

```
String[] myCommandCategories = {"app.contacts", "action.view"};
metadata.setCommandCategories(new CategoryCollection(myCommandCategories));
String[] myContextCategories = {"emailaddr"};
metadata.setContextCategories(new CategoryCollection(myContextCategories));
```

6. Registrieren Sie den Befehl.
 - a. Wenn Ihr Befehl nur in Ihrer Anwendung verfügbar sein soll, verwenden Sie ein `LocalCommandRegistrarConnection`-Objekt. Die lokale Registry ist nur im aktuellen Prozess verfügbar.

```
LocalCommandRegistrarConnection connection =
    new LocalCommandRegistrarConnection();
connection.registerCommand(new ViewContactCommand(), metadata);
```

Anmerkung: Um den Anwendungen zu ermöglichen, Instanzen des Befehls dynamisch zu erstellen, müssen Sie den Befehl nur mit den Metadaten durch Aufrufen von `registerCommand(CommandMetadata)` registrieren. Andernfalls wird der registrierte `CommandHandler` verwendet.

- b. Wenn Ihr Befehl in beliebigen Anwendungen verfügbar sein soll, verwenden Sie ein `RemoteCommandRegistrarConnection`-Objekt. Die globale Registry steht allen Prozessen zur Verfügung, die auf dem Gerät ausgeführt werden.

```
RemoteCommandRegistrarConnection connection =
    new RemoteCommandRegistrarConnection();
connection.registerCommand(new ViewContactCommand(), metadata);
```

7. Rufen Sie den Befehl von einer beliebigen Position in Ihrer Anwendung (bei lokalen Befehlen) oder von einer beliebigen Anwendung auf dem Gerät (bei globalen Befehlen) auf und verwenden Sie ihn.
 - a. Erstellen Sie ein `CommandRequest`-Objekt und füllen Sie es mit Informationen zu dem Befehl, den Sie aus der Registry abrufen möchten.

```
CommandRequest request = new CommandRequest();
String[] myCommandCategories = {"app.contacts", "action.view"};
request.setCommandCategories(new CategoryCollection(myCommandCategories));
...
```

- b. Rufen Sie den Befehl aus der Registry ab.

```
// using the local registry
LocalCommandRegistrarConnection connection =
    new LocalCommandRegistrarConnection();
Command command = connection.getCommand(request);
// using the global registry
RemoteCommandRegistrarConnection connection =
    new RemoteCommandRegistrarConnection();
Command command = connection.getCommand(request);
```

- c. Verwenden Sie den Befehl. Das folgende Beispiel führt den Befehl in Abhängigkeit vom Kontext aus.

```
command.execute(context); // context is the context object
```

Codebeispiele

Zwei Beispielanwendungen, die das Command Framework API verwenden, sind im BlackBerry® Java® SDK enthalten:

- Command Framework Demo Remote App veranschaulicht, wie ein Befehl erstellt und in der globalen Registry zur Verwendung in anderen Anwendungen gespeichert wird.
- Command Framework Demo veranschaulicht, wie Befehle erstellt, in der lokalen Registry gespeichert und als Menüeinträge in Popup-Menüs für Felder in Abhängigkeit zum Feldinhalt hinzugefügt werden. Diese Beispielanwendung führt außerdem einen Befehl aus, der von der Beispielanwendung Command Framework Demo Remote App in der globalen Registry gespeichert wird.

Anordnen von Benutzeroberflächenkomponenten

7

Sie können die Benutzeroberflächenkomponenten auf einem Anwendungsbildschirm mit dem BlackBerry® Layout-Manager-API anordnen. Die folgenden Klassen erweitern die im `net.rim.device.apu.ui`-Paket bereitgestellte Manager-Klasse und stellen vordefinierte Layouts für die Benutzeroberflächenkomponenten auf dem Bildschirm Ihrer Anwendung bereit.

Layout-Manager	Beschreibung
<code>FlowFieldManager</code>	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten in Abhängigkeit von der Größe des Bildschirms vertikal und anschließend horizontal an. Die erste Benutzeroberflächenkomponente wird in der linken oberen Ecke des Bildschirms platziert und die folgenden Komponenten werden horizontal rechts von der ersten Komponente platziert, bis der Bildschirm über die gesamte Breite gefüllt ist. Wenn keine weiteren Benutzeroberflächenkomponenten in die erste Zeile passen, wird die nächste Benutzeroberflächenkomponente unter die erste Zeile von Komponenten in einer Zeile platziert, deren Höhe der Höhe der größten Komponente in der Zeile darüber entspricht. Sie können vertikale Stile anwenden (zum Beispiel <code>Field.FIELD_TOP</code> , <code>Field.FIELD_BOTTOM</code> oder <code>Field.FIELD_VCENTER</code>), um die Benutzeroberflächenkomponenten innerhalb ihrer Zeile vertikal auszurichten.
<code>HorizontalFieldManager</code>	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten in einer einzelnen horizontalen Zeile vom linken zum rechten Rand des Bildschirms an. Da dieser Layout-Manager Benutzeroberflächenkomponenten horizontal anordnet, können Sie keine horizontalen Stile an Benutzeroberflächenkomponenten (zum Beispiel <code>Field.FIELD_LEFT</code> , <code>Field.FIELD_HCENTER</code> oder <code>Field.FIELD_RIGHT</code>) anwenden. Sie können vertikale Stile (zum Beispiel <code>Field.FIELD_TOP</code> , <code>Field.FIELD_BOTTOM</code> oder <code>Field.FIELD_VCENTER</code>) anwenden. Wenn die Benutzeroberflächenkomponenten nicht in die verfügbare Breite des Bildschirms passen, sollten Sie den <code>Manager.HORIZONTAL_SCROLL</code> -Stil verwenden, um einen horizontalen Bildlauf zu aktivieren. Anderenfalls werden auf dem Bildschirm so viele Benutzeroberflächenkomponenten wie innerhalb der verfügbaren Bildschirmbreite möglich angezeigt, wobei die übrigen entfallen. Die Benutzeroberflächenkomponenten sind vorhanden, aber nicht sichtbar. Dies kann zu einem unerwarteten Verhalten beim Scrollen führen.
<code>VerticalFieldManager</code>	Dieser Layout-Manager ordnet Benutzeroberflächenkomponenten auf dem Bildschirm von oben nach unten in einer einzelnen vertikalen Spalte an. Da dieser Layout-Manager Elemente vertikal anordnet, können Sie keine

Layout-Manager	Beschreibung
	vertikalen Stile an Benutzeroberflächenkomponenten (zum Beispiel <code>Field.FIELD_TOP</code> , <code>Field.FIELD_BOTTOM</code> oder <code>Field.FIELD_VCENTER</code>) anwenden. Sie können horizontale Stile (zum Beispiel <code>Field.FIELD_LEFT</code> , <code>Field.FIELD_HCENTER</code> oder <code>Field.FIELD_RIGHT</code>) anwenden.

Sie können zusätzliche Layout-Manager verwenden, um Benutzeroberflächenkomponenten in Ihrer Anwendung anzuordnen. Zum Beispiel können Sie den `GridFieldManager`-Layout-Manager verwenden, um Benutzeroberflächenkomponenten auf einem Bildschirm in Zeilen und Spalten anzuordnen und dadurch ein Raster zu erstellen. Sie können den `EyelidFieldManager`-Layout-Manager verwenden, um Benutzeroberflächenkomponenten in einem Paar von Managern anzuzeigen, die vorübergehend oben und unten auf dem Bildschirm angezeigt werden.

Anordnen von Benutzeroberflächenkomponenten

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
net.rim.device.api.ui.container.HorizontalFieldManager;
net.rim.device.api.ui.component.ButtonField;
```

2. Erstellen Sie eine Instanz von `HorizontalFieldManager`.

```
HorizontalFieldManager _fieldManagerBottom = new HorizontalFieldManager();
```

3. Rufen Sie `add()` auf, um dem Bildschirm `HorizontalFieldManager` hinzuzufügen.

```
myScreen.add(_fieldManagerBottom);
```

4. Erstellen Sie eine Instanz von `ButtonField`.

```
ButtonField mySubmitButton = new ButtonField("Submit");
```

5. Fügen Sie das `ButtonField` zu `HorizontalFieldManager` hinzu.

```
_fieldManagerBottom.add(mySubmitButton);
```

Erstellen eines Raster-Layouts

Anmerkung: Für Informationen zum Erstellen einer Gittergröße in der BlackBerry® Java® Development Environment vor Version 5.0 besuchen Sie die Seite <http://www.blackberry.com/knowledgecenterpublic>, und lesen Sie Artikel DB-00783.

Sie können mithilfe der `GridFieldManager`-Klasse Felder auf einem Bildschirm in Zeilen und Spalten anordnen, um ein Raster zu erstellen. Wenn Sie ein Raster erstellen, können Sie die Anzahl der Zeilen und Spalten angeben. Nach dem Erstellen eines Rasters können Sie die Anzahl der Zeilen und Spalten, die es enthält, nicht mehr ändern.

Der Index von Rastern beginnt bei null. Deshalb befindet sich die erste Zelle bei Zeile 0, Spalte 0. In einem Gebietsschema mit einer Textrichtung von links nach rechts befindet sich die erste Zelle in der linken oberen Ecke des Rasters.

Left-to-right locale		
This is cell (0,0)	This is cell (0,1)	This is cell (0,2)
This is cell (1,0)	This is cell (1,1)	This is cell (1,2)

In einem Gebietsschema mit einer Textrichtung von rechts nach links befindet sich die erste Zelle in der rechten oberen Ecke des Rasters.

Right-to-left locale		
This is cell (0,2)	This is cell (0,1)	This is cell (0,0)
This is cell (1,2)	This is cell (1,1)	This is cell (1,0)

Sie können einem Raster Felder sequenziell hinzufügen (von links nach rechts oder von oben nach unten in Gebietsschemas mit einer Textrichtung von links nach rechts; von rechts nach links oder von oben nach unten in Gebietsschemas mit einer Textrichtung von rechts nach links) oder durch Angeben einer Zeile und einer Spalte im Raster. Sie können Felder löschen, Felder einfügen, die Abstände zwischen Spalten und Zeilen angeben und die Eigenschaften eines Rasters abrufen.

Raster haben keine definierten Überschriftenzeilen oder Überschriftenspalten. Sie können die Darstellung von Überschriften emulieren, indem Sie die Darstellung der Felder in der ersten Zeile oder der ersten Spalte des Rasters ändern.

Sie können horizontal und vertikal durch Raster scrollen, wenn die Breite oder Höhe des Rasters über den Bildschirmbereich hinausgeht.

Sie können durch Aufrufen von `GridFieldManager.setColumnProperty()` die Spaltenbreite und durch Aufrufen von `GridFieldManager.setRowProperty()` die Zeilenhöhe angeben. Wenn Sie diese Methoden aufrufen, müssen Sie eine `GridFieldManager`-Eigenschaft angeben.

Eigenschaft	Beschreibung
<code>FIXED_SIZE</code>	Breite oder Höhe ist ein fester Wert in Pixel

Eigenschaft	Beschreibung
PREFERRED_SIZE	Breite oder Höhe ist eine bevorzugte Größe basierend auf der maximalen bevorzugten Größe der Felder in der Spalte oder der Zeile (PREFERRED_SIZE ist die Standardeigenschaft)
PREFERRED_SIZE_WITH_MAXIMUM	Breite oder Höhe ist eine bevorzugte Größe bis zu einer maximalen Größe
AUTO_SIZE	Breite oder Höhe basiert auf dem verfügbaren Platz auf dem Bildschirm

Erstellen einer Gittergröße

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Im Beispiel stellt die `GridScreen`-Klasse, beschrieben in Schritt 3, den benutzerdefinierten Bildschirm dar.

```
class GridFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        GridFieldManagerDemo theApp = new GridFieldManagerDemo();
        theApp.enterEventDispatcher();
    }
    GridFieldManagerDemo()
    {
        pushScreen(new GridScreen());
    }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class GridScreen extends MainScreen
{
    public GridScreen()
    {
    }
}
```

4. Im Bildschirmkonstruktor rufen Sie `setTitle()` auf, um den Text festzulegen, den Sie im Titelausschnitt vom Bildschirm anzeigen wollen.

```
setTitle("GridFieldManager Demo");
```

- Erstellen Sie im Bildschirmkonstruktor eine Instanz der `GridFieldManager`-Klasse. Geben Sie die Anzahl der Reihen, die Anzahl der Spalten und den Stil des Rasters an (mithilfe eines Stils, der von `net.rim.device.api.ui.Manager` geerbt wird). Geben Sie 0 für den Stil an, um den Standardstil zu verwenden.

```
GridFieldManager grid;
grid = new GridFieldManager(2,3,0);
```

- Rufen Sie im Bildschirmkonstruktor `GridFieldManager.add()` auf, um Felder zum Raster hinzuzufügen.

```
grid.add(new LabelField("one"));
grid.add(new LabelField("two"));
grid.add(new LabelField("three"));
grid.add(new LabelField("four"));
grid.add(new LabelField("five"));
```

- Rufen Sie im Bildschirmkonstruktor die `GridFieldManager.set()`-Methoden auf, um die Eigenschaften des Rasters anzugeben.

```
grid.setColumnPadding(20);
grid.setRowPadding(20);
```

- Rufen Sie `Screen.add()` im Bildschirmkonstruktor auf, um dem Bildschirm das Raster hinzuzufügen.

```
add(grid);
```

Nach Abschluss erforderliche Aktion:

Sie können das Raster ändern, nachdem Sie es erstellt haben. Zum Beispiel können Sie Felder hinzufügen, Felder löschen oder die Eigenschaften des Rasters ändern.

Codebeispiel: Erstellen eines Raster-Layouts

```
/*
 * GridFieldManagerDemo.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2009
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class GridFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        GridFieldManagerDemo theApp = new GridFieldManagerDemo();
        theApp.enterEventDispatcher();
    }
    GridFieldManagerDemo()
    {
        pushScreen(new GridScreen());
    }
}
class GridScreen extends MainScreen
```

```

{
    public GridScreen()
    {
        setTitle("GridFieldManager Demo");
        GridFieldManager grid = new GridFieldManager(2,3,0);
        grid.add(new LabelField("one"));
        grid.add(new LabelField("two"));
        grid.add(new LabelField("three"));
        grid.add(new LabelField("four"));
        grid.add(new LabelField("five"));
        grid.setColumnPadding(20);
        grid.setRowPadding(20);
        add(grid);
        // The grid looks like this:
        // | one | two | three
        // | four | five |
        // insert a cell first row, second column
        grid.insert(new LabelField("insert"), 0, 1);
        // The grid now looks like this:
        // | one | insert | two
        // | three | four | five
        // delete a cell second row, second column
        grid.delete(1,1);
        // The grid now looks like this:
        // | one | insert | two
        // | three | | five
        // Add field to first unoccupied cell
        grid.add(new LabelField("new"));
        // The grid now looks like this:
        // | one | insert | two
        // | three | new | five
    }
}

```

Anzeigen von Feldern in einem Paar temporärer Manager

Sie können die `EyelidFieldManager`-Klasse verwenden, um Felder in einem Paar von Managern anzuzeigen, die vorübergehend oben und unten auf dem Bildschirm angezeigt werden.

Standardmäßig werden die Felder angezeigt, wenn der BlackBerry®-Gerätebenutzer den Trackball bewegt oder, bei einem Gerät mit Touchscreen, den Bildschirm berührt. Die Felder werden nach einem bestimmten Zeitraum der Inaktivität (standardmäßig 1,2 Sekunden) ausgeblendet. Sie können diese Standardeigenschaften außer Kraft setzen.

Es gibt keine Beschränkung für die Anzahl und die Größe der Felder. Wenn die Manager mehr Felder enthalten, als auf den Bildschirm passen, überlappen der obere und der untere Manager einander, wobei sich der obere Manager im Vordergrund befindet.

Vorübergehendes Anzeigen eines `ButtonField` und eines `LabelField` oben bzw. unten auf dem Bildschirm

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.extension.container.*;
```

- Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `EyelidFieldManagerDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class EyelidFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        EyelidFieldManagerDemo app = new EyelidFieldManagerDemo();
        app.enterEventDispatcher();
    }
    public EyelidFieldManagerDemo()
    {
        pushScreen(new EyelidFieldManagerDemoScreen());
    }
}
```

- Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class EyelidFieldManagerDemoScreen extends MainScreen {
{
    public EyelidFieldManagerDemoScreen()
    {
    }
}
```

- Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird.

```
setTitle("EyelidFieldManager Demo");
```

- Erstellen Sie im Bildschirmkonstruktor eine Instanz der `EyelidFieldManager`-Klasse.

```
EyelidFieldManager manager = new EyelidFieldManager();
```

- Rufen Sie im Bildschirmkonstruktor `EyelidFieldManager.addTop()` auf, um dem obersten Manager des `EyelidFieldManager` ein `LabelField`-Objekt hinzuzufügen.

```
manager.addTop(new LabelField("Hello World"));
```

- Erstellen Sie im Bildschirmkonstruktor ein `HorizontalFieldManager`-Objekt. Rufen Sie `HorizontalFieldManager.add()` auf, um dem `HorizontalFieldManager` Schaltflächen hinzuzufügen. Rufen Sie `EyelidFieldManager.addBottom()` auf, um den `HorizontalFieldManager` zum untersten Manager des `EyelidFieldManager` hinzuzufügen.

```
HorizontalFieldManager buttonPanel = new HorizontalFieldManager
(Field.FIELD_HCENTER | Field.USE_ALL_WIDTH);
buttonPanel.add(new SimpleButton("Button 1"));
buttonPanel.add(new SimpleButton("Button 2"));
manager.addBottom(buttonPanel);
```

8. Rufen Sie im Bildschirmkonstruktor `EyelidFieldManager.setEyelidDisplayTime()` auf, um den Zeitraum der Inaktivität in Millisekunden festzulegen, der vergehen muss, bevor das Manager-Paar ausgeblendet wird.

```
manager.setEyelidDisplayTime(3000);
```

9. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm `EyelidFieldManager` hinzuzufügen.

```
add(manager);
```

Codebeispiel: Vorübergehendes Anzeigen eines `ButtonField` und eines `LabelField` oben bzw. unten am Bildschirm

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.extension.container.*;
public class EyelidFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        EyelidFieldManagerDemo app = new EyelidFieldManagerDemo();
        app.enterEventDispatcher();
    }
    public EyelidFieldManagerDemo()
    {
        pushScreen(new EyelidFieldManagerDemoScreen());
    }
}
class EyelidFieldManagerDemoScreen extends MainScreen
{
    public EyelidFieldManagerDemoScreen()
    {
        setTitle("EyelidFieldManager Demo");
        EyelidFieldManager manager = new EyelidFieldManager();
        manager.addTop(new LabelField("Hello World"));
        HorizontalFieldManager buttonPanel = new HorizontalFieldManager
(Field.FIELD_HCENTER | Field.USE_ALL_WIDTH);
        buttonPanel.add(new ButtonField("Button 1"));
        buttonPanel.add(new ButtonField("Button 2"));
        manager.addBottom(buttonPanel);
        manager.setEyelidDisplayTime(3000);
        add(manager);
    }
}
```

Anzeigen eines Felds an einer absoluten Position auf dem Bildschirm

Sie können die `AbsoluteFieldManager`-Klasse verwenden, um basierend auf den XY-Koordinaten eine absolute Feldposition (im Gegensatz zu einer Position, die relativ zu anderen Feldern ist) auf dem Bildschirm festzulegen.

Sie können die `AbsoluteFieldManager`-Klasse verwenden, um auf dem Bildschirm die Platzierung und das Überlappen von Feldern zu steuern.

Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `AbsoluteFieldManagerDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class AbsoluteFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        AbsoluteFieldManagerDemo app = new AbsoluteFieldManagerDemo();
        app.enterEventDispatcher();
    }
    public AbsoluteFieldManagerDemo()
    {
        pushScreen(new AbsoluteFieldManagerDemoScreen());
    }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse. Rufen Sie `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird.

```
class AbsoluteFieldManagerDemoScreen extends MainScreen
{
    public AbsoluteFieldManagerDemoScreen()
    {
        setTitle("AbsoluteFieldManager Demo");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `AbsoluteFieldManager`-Klasse.

```
AbsoluteFieldManager manager = new AbsoluteFieldManager();
```

5. Erstellen und initialisieren Sie im Bildschirmkonstruktor eine Variable, um die Y-Koordinate zu speichern, die der vertikalen Mitte des Bildschirms entspricht.

```
int halfwayDown = Display.getHeight() / 2;
```

6. Rufen Sie im Bildschirmkonstruktor `AbsoluteFieldManager.add()` auf, um an der vertikalen Mitte des Bildschirms ein `LabelField`-Objekt mit einem X-Koordinatenwert von 2 hinzuzufügen.

```
manager.add(new LabelField("Hello world"), 2, halfwayDown);
```

7. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm `AbsoluteFieldManager` hinzuzufügen.

```
add(manager);
```

Codebeispiel: Anzeigen einer Bezeichnung an einer absoluten Bildschirmposition

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
public class AbsoluteFieldManagerDemo extends UiApplication
{
    public static void main(String[] args)
    {
        AbsoluteFieldManagerDemo app = new AbsoluteFieldManagerDemo();
        app.enterEventDispatcher();
    }
    public AbsoluteFieldManagerDemo()
    {
        pushScreen(new AbsoluteFieldManagerDemoScreen());
    }
}
class AbsoluteFieldManagerDemoScreen extends MainScreen
{
    public AbsoluteFieldManagerDemoScreen()
    {
        setTitle("AbsoluteFieldManager Demo");
        AbsoluteFieldManager manager = new AbsoluteFieldManager();
        int halfwayDown = Display.getHeight() / 2;
        manager.add(new LabelField("Hello world"), 2, halfwayDown);
        add(manager);
    }
}
```

Benutzeroberflächen-Komponenten

Hinzufügen einer Benutzeroberflächenkomponente zu einem Bildschirm

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.component.CheckBoxField`
 - `net.rim.device.api.ui.container.MainScreen`
2. Erstellen Sie eine Instanz einer Benutzeroberflächenkomponente.

```
CheckBoxField myCheckbox = new CheckBoxField("First checkbox", true);
```
3. Fügen Sie die Benutzeroberflächenkomponente Ihrer Erweiterung einer Screen-Klasse hinzu.

```
mainScreen.add(myCheckbox);
```

Ausrichten eines Feldes an einer Textzeile

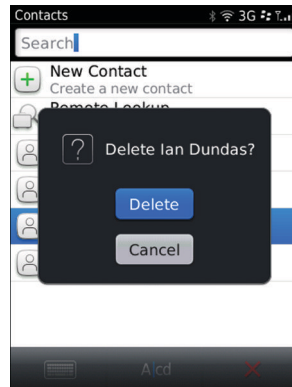
Sie können eine Anwendung erstellen, die ein `Field`-Objekt mithilfe der Markierung `Field.FIELD_LEADING` am Anfang einer Textzeile ausrichten kann. Wenn Sie zum Beispiel ein `Field` mit der Ausrichtungsmarkierung `Field.FIELD_LEADING` erstellen und einem `VerticalFieldManager` das `Field` hinzufügen, wird das `Field`, wenn die Anwendung beispielsweise beginnt, ein englisches oder chinesisches Gebietsschema zu verwenden, an der linken Seite des Bildschirms ausgerichtet. Wenn die Anwendung beginnt, ein arabisches oder hebräisches Gebietsschema zu verwenden, wird das `Field` an der rechten Seite des Bildschirms ausgerichtet.

Schaltflächen

Verwenden Sie Schaltflächen, um Benutzern zu ermöglichen, eine Aktion über ein Dialogfeld auszuführen. Menüs enthalten in der Regel Aktionen, die mit einem Bildschirm verknüpft sind.

Benutzer können über eine Schaltfläche die folgenden Aktionen ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Markieren einer Schaltfläche	Mit dem Finger über das Trackpad fahren	• Leichtes Berühren der Schaltfläche • Mit dem Finger über das Trackpad fahren
Ausführen einer Aktion	Klicken mit dem Trackpad oder Drücken der Eingabetaste	• Tippen auf das Element • Klicken mit dem Trackpad. • Drücken Sie die Eingabetaste.



Bewährtes Verfahren: Implementieren von Schaltflächen

- Vermeiden Sie auf einem Anwendungsbildschirm den Einsatz von Schaltflächen. Zur Bereitstellung von Aktionen, die mit einem Bildschirm verknüpft sind, verwenden Sie stattdessen das Anwendungsmenü. Auf BlackBerry®-Geräten mit Trackpad steht den Benutzern das Menü sofort zur Verfügung, und das unabhängig davon, an welcher Position sich der Cursor auf dem Bildschirm befindet. Schaltflächen sind statisch und erfordern, dass sie von den Benutzern hervorgehoben werden, um die mit der Schaltfläche verknüpfte Aktion auszuführen. Wenn Sie Schaltflächen verwenden, nehmen Sie ebenfalls Menüelemente für die Aktionen im Anwendungsmenü auf. Auf BlackBerry-Geräten mit Touchscreen können Sie Schaltflächen für kritische Aktionen verwenden.
- Verwenden Sie Kontrollkästchen für Optionen, z. B. zum Ein- und Ausschalten einer Funktion.
- Verwenden Sie die `ButtonField`-Klasse, um Schaltflächen zu erstellen.
- Verwenden Sie für die Standardschaltfläche jene Schaltfläche, die Benutzer am wahrscheinlichsten anklicken. Vermeiden Sie die Verwendung einer Schaltfläche, die mit einer beendenden Aktion verknüpft ist, als Standardschaltfläche.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen.
- Verwenden Sie nach Möglichkeit Bezeichnungen, die aus einem Wort bestehen. Die Größe einer Schaltfläche ändert sich mit der Länge der Bezeichnung. Wenn eine Bezeichnung zu lang ist, zeigt ein Auslassungspunkt (...) an, dass der Text abgeschnitten ist.
- Verwenden Sie für Bezeichnungen Verben, die die verknüpfte Aktion beschreiben (zum Beispiel "Abbrechen", "Löschen", "Verwerfen" oder "Speichern"). Falls erforderlich, fügen Sie weitere Beschreibungen anderswo auf dem Bildschirm hinzu (zum Beispiel in einer Anwendungsnachricht).
- Vermeiden Sie die Verwendung der Bezeichnungen "Ja" und "Nein".
- Vermeiden Sie Interpunktion in einer Bezeichnung. Verwenden Sie in der Bezeichnung einer Schaltfläche Auslassungspunkte um anzuzeigen, dass Benutzer eine weitere Aktion ausführen müssen, nachdem sie auf die Schaltfläche klicken.

Erstellen einer Schaltfläche

1. Importieren Sie die `net.rim.device.api.ui.component.ButtonField`-Klasse.
2. Erstellen Sie mithilfe eines Stilparameters eine Instanz eines `ButtonField`-Objekts.

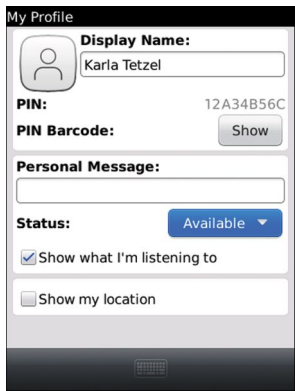
```
ButtonField mySubmitButton = new ButtonField("Submit");
```

Kontrollkästchen

Verwenden Sie Kontrollkästchen für Binäroptionen, die Benutzer problemlos verstehen können. Verwenden Sie zum Beispiel ein Kontrollkästchen für eine Option, die ein- und ausgeschaltet werden kann.

Benutzer können mit einem Kontrollkästchen die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Auswählen eines Kontrollkästchens	Drücken der Leertaste oder Klicken mit dem Trackpad.	• Tippen auf das Element • Drücken Sie die Leertaste. • Klicken mit dem Trackpad.



Bewährtes Verfahren: Implementieren von Kontrollkästchen

- Verwenden Sie Kontrollkästchen, wenn Benutzer mehrere Optionen auswählen können.
- Verwenden Sie die `CheckboxField`-Klasse, um Kontrollkästchen zu erstellen.
- Starten Sie keine Aktion, wenn Benutzer ein Kontrollkästchen auswählen. Öffnen Sie zum Beispiel keinen neuen Bildschirm.
- Richten Sie Kontrollkästchen senkrecht aus.

- Gruppieren und ordnen Sie Kontrollkästchen logisch an (gruppieren Sie beispielsweise zusammen gehörende Objekte oder führen Sie die häufigsten Objekte zuerst an). Vermeiden Sie eine alphabetische Sortierung der Kontrollkästchen; die alphabetische Reihenfolge ist sprachspezifisch.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen. Überprüfen Sie, ob die Bezeichnung deutlich beschreibt, was geschieht, wenn Benutzer das Kontrollkästchen aktivieren.
- Verwenden Sie nach Möglichkeit positive Bezeichnungen. Wenn beispielsweise Benutzer die Option zum Aktivieren oder Deaktivieren einer Funktion haben, verwenden Sie in der Bezeichnung "Aktivieren" statt "Deaktivieren".
- Platzieren Sie die Bezeichnungen auf der rechten Seite der Kontrollkästchen. Bei "Optionen" platzieren Sie die Bezeichnungen auf der linken Seite von Kontrollkästchen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Interpunktieren Sie Bezeichnungen für Kontrollkästchen mit einem Doppelpunkt (:).

Erstellen eines Kontrollkästchens

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
```

```
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor ein Kontrollkästchen mithilfe der `CheckBoxField`-Klasse. Geben Sie im `CheckBoxField`-Konstruktor die Bezeichnung für das Kontrollkästchen an, und verwenden Sie einen booleschen Wert, um anzuzeigen, ob das Kontrollkästchen die Standardauswahl ist (`true` zeigt zum Beispiel an, dass das Kontrollkästchen standardmäßig ausgewählt ist). Rufen Sie `add()` auf, um dem Bildschirm das Kontrollkästchen hinzuzufügen.

```
add(new CheckBoxField("First Check Box", true));
add(new CheckBoxField("Second Check Box", false));
```

5. Um das Standardverhalten eines Kontrollkästchens zu ändern, verwenden Sie die Stile, die von der `Field`-Klasse geerbt wurden. Um zum Beispiel ein Kontrollkästchen zu erstellen, das die Benutzer nicht ändern können, verwenden Sie den `READONLY`-Stil.

```
add(new CheckBoxField("First Check Box", true, this.READONLY));
```

6. Um die Standardfunktionalität zu überschreiben, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, überschreiben Sie in der Erweiterung der `MainScreen`-Klasse die Methode `MainScreen.onSavePrompt()`. Im folgenden Codebeispiel wird `true` zurückgegeben. Dies bedeutet, dass vor dem Schließen keine Benutzeraufforderung durch die Anwendung erfolgt.

```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen eines Kontrollkästchens

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        add(new CheckBoxField("First Check Box", true));
        add(new CheckBoxField("Second Check Box", false));
    }
}
```

```
public boolean onSavePrompt()
{
    return true;
}
```

Erstellen einer Bitmap

1. Importieren Sie die `net.rim.device.api.ui.component.BitmapField`-Klasse.
2. Erstellen Sie eine Instanz von `BitmapField`.

```
BitmapField myBitmapField = new BitmapField();
```

Erstellen eines benutzerdefinierten Feldes

Sie können einem benutzerdefinierten Feld nur benutzerdefinierte Kontextmenüeinträge und benutzerdefinierte Layouts hinzufügen.

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.Field`
 - `java.lang.String`
 - `net.rim.device.api.ui.Font`
 - `java.lang.Math`
 - `net.rim.device.api.ui.Graphics`
2. Importieren Sie die `net.rim.device.api.ui.DrawStyle`-Schnittstelle.
3. Erweitern Sie die `Field`-Klasse oder eine ihrer Unterklassen durch Implementieren der `DrawStyle`-Schnittstelle, um die Eigenschaften des benutzerdefinierten Feldes anzugeben und die Zeichnungsstile zu aktivieren.

```
public class CustomButtonField extends Field implements DrawStyle {
    public static final int RECTANGLE = 1;
    public static final int TRIANGLE = 2;
    public static final int OCTAGON = 3;
    private String _label;
    private int _shape;
    private Font _font;
    private int _labelHeight;
    private int _labelWidth;
}
```

4. Implementieren Sie Konstruktoren, um die Bezeichnung, die Form und den Stil der benutzerdefinierten Schaltfläche zu definieren.

```
public CustomButtonField(String label) {
    this(label, RECTANGLE, 0);
}
public CustomButtonField(String label, int shape) {
    this(label, shape, 0);
}
```

```

}
public CustomButtonField(String label, long style) {
    this(label, RECTANGLE, style);
}
public CustomButtonField(String label, int shape, long style) {
    super(style);
    _label = label;
    _shape = shape;
    _font = getFont();
    _labelHeight = _font.getHeight();
    _labelWidth = _font.getAdvance(_label);
}

```

5. Implementieren Sie `layout()`, um die Anordnung der Felddaten anzugeben. Führen Sie die komplexesten Berechnungen in `layout()` statt in `paint()` aus. Der Manager des Feldes ruft `layout()` auf, um zu bestimmen, wie das Feld seine Inhalte im verfügbaren Platz anordnet. Rufen Sie `Math.min()` auf, um die kleineren Werte für die angegebene Breite und Höhe sowie die bevorzugte Breite und Höhe des Feldes zurückzugeben. Rufen Sie `Field.setExtent(int, int)` auf, um die erforderlichen Dimensionen für das Feld festzulegen.

```

protected void layout(int width, int height) {
    _font = getFont();
    _labelHeight = _font.getHeight();
    _labelWidth = _font.getAdvance(_label);
    width = Math.min( width, getPreferredWidth() );
    height = Math.min( height, getPreferredHeight() );
    setExtent( width, height );
}

```

6. Implementieren Sie `getPreferredWidth()` mithilfe der relativen Maßangaben der Feldbezeichnung, um sicherzustellen, dass die Bezeichnung die Abmessungen der Komponente nicht überschreitet. Verwenden Sie einen switch-Block, um die bevorzugte Breite basierend auf der Form des benutzerdefinierten Feldes zu bestimmen. Verwenden Sie für jeden Formtyp eine if-Anweisung, um Maßangaben zu vergleichen und die bevorzugte Breite für das benutzerdefinierte Feld zu bestimmen.

```

public int getPreferredWidth() {
    switch(_shape) {
        case TRIANGLE:
            if (_labelWidth < _labelHeight) {
                return _labelHeight << 2;
            } else {
                return _labelWidth << 1;
            }
        case OCTAGON:
            if (_labelWidth < _labelHeight) {
                return _labelHeight + 4;
            } else {
                return _labelWidth + 8;
            }
        case RECTANGLE: default:
            return _labelWidth + 8;
    }
}

```

7. Implementieren Sie `getPreferredHeight()` mithilfe der relativen Maßangaben der Feldbezeichnung, um die bevorzugte Höhe zu bestimmen. Verwenden Sie einen `switch`-Block, um die bevorzugte Höhe basierend auf der Form des benutzerdefinierten Feldes zu bestimmen. Verwenden Sie für jeden Formtyp eine `if`-Anweisung, um Maßangaben zu vergleichen und die bevorzugte Höhe für das benutzerdefinierte Feld zu bestimmen.

```
public int getPreferredHeight() {
    switch(_shape) {
        case TRIANGLE:
            if (_labelWidth < _labelHeight) {
                return _labelHeight << 1;
            } else {
                return _labelWidth;
            }
        case RECTANGLE:
            return _labelHeight + 4;
        case OCTAGON:
            return getPreferredWidth();
    }
    return 0;
}
```

8. Implementieren Sie `paint()`. Der Manager eines Feldes ruft `paint()` auf, um das Feld neu zu zeichnen, wenn ein Bereich des Feldes als ungültig markiert wird. Verwenden Sie einen `switch`-Block, um das Feld basierend auf der Form des benutzerdefinierten Feldes neu zu zeichnen. Verwenden bei einem Feld mit einer dreieckigen oder achteckigen Form die Breite des Feldes, um die horizontale und vertikale Position des Anfangs- und Endpunkts einer Linie zu berechnen. Rufen Sie `graphics.drawLine()` auf, und verwenden Sie die Anfangs- und Endpunkte, um die Linien zu zeichnen, die das benutzerdefinierte Feld definieren. Rufen Sie bei einem Feld mit einer rechteckigen Form `graphics.drawRect()` auf, und verwenden Sie die Breite und die Höhe des Feldes, um das benutzerdefinierte Feld zu zeichnen. Rufen Sie `graphics.drawText()` auf, und verwenden Sie die Breite des Feldes, um eine Textzeichenfolge in einen Bereich des Feldes zu zeichnen.

```
protected void paint(Graphics graphics) {
    int textX, textY, textWidth;
    int w = getWidth();
    switch(_shape) {
        case TRIANGLE:
            int h = (w>>1);
            int m = (w>>1)-1;
            graphics.drawLine(0, h-1, m, 0);
            graphics.drawLine(m, 0, w-1, h-1);
            graphics.drawLine(0, h-1, w-1, h-1);
            textWidth = Math.min(_labelWidth,h);
            textX = (w - textWidth) >> 1;
            textY = h >> 1;
            break;
        case OCTAGON:
            int x = 5*w/17;
            int x2 = w-x-1;
            int x3 = w-1;
            graphics.drawLine(0, x, 0, x2);
            graphics.drawLine(x3, x, x3, x2);
            graphics.drawLine(x, 0, x2, 0);
            graphics.drawLine(x, x3, x2, x3);
    }
}
```

```

        graphics.drawLine(0, x, x, 0);
        graphics.drawLine(0, x2, x, x3);
        graphics.drawLine(x2, 0, x3, x);
        graphics.drawLine(x2, x3, x3, x2);
        textWidth = Math.min(_labelWidth, w - 6);
        textX = (w-textWidth) >> 1;
        textY = (w-_labelHeight) >> 1;
        break;
    case RECTANGLE: default:
        graphics.drawRect(0, 0, w, getHeight());
        textX = 4;
        textY = 2;
        textWidth = w - 6;
        break;
    }
    graphics.drawText(_label, textX, textY, (int)(getStyle() & DrawStyle.ELLIPSIS
| DrawStyle.HALIGN_MASK ), textWidth );
}

```

9. Implementieren Sie die Field `set()`- und `get()`-Methoden. Implementieren Sie die Field `getLabel()`-, Field `getShape()`-Field `setLabel(String label)`-und Field `setShape(int shape)`-Methoden, um die Instanzvariablen des benutzerdefinierten Felds zurückzugeben.

```

public String getLabel() {
    return _label;
}
public int getShape() {
    return _shape;
}
public void setLabel(String label) {
    _label = label;
    _labelWidth = _font.getAdvance(_label);
    updateLayout();
}
public void setShape(int shape) {
    _shape = shape;
}

```

Erstellen eines Felds, um Webinhalte anzuzeigen

Anzeigen von HTML-Inhalten in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;

```

- Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
```

- Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
    }
}
```

- Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

```
BrowserField myBrowserField = new BrowserField();
```

- Rufen Sie `add()` im Bildschirmkonstruktor auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

```
add(myBrowserField);
```

- Rufen Sie `BrowserField.displayContent()` im Bildschirmkonstruktor auf, um den HTML-Inhalt festzulegen und anzuzeigen.

```
myBrowserField.displayContent("<html><body><h1>Hello World!</h1></body></html>", "http://localhost");
```

Codebeispiel: Anzeigen von HTML-Inhalt in einem Browserfeld

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
```

```

    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.displayContent("<html><body><h1>Hello
World!</h1></body></html>", "http://localhost");
    }
}

```

Anzeigen von HTML-Inhalten von einer Webseite in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;

```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```

public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}

```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```

class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
    }
}

```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserField`-Klasse.

```
BrowserField myBrowserField = new BrowserField();
```

5. Rufen Sie `add()` im Bildschirmkonstruktor auf, um dem Bildschirm das `BrowserField`-Objekt hinzuzufügen.

```
add(myBrowserField);
```

6. Rufen Sie `BrowserField.requestContent()` im Bildschirmkonstruktor auf, um den Ort des HTML-Inhalts festzulegen und anzuzeigen.

```
myBrowserField.requestContent("http://www.blackberry.com");
```

Codebeispiel: Anzeigen von HTML-Inhalt von einer Webseite in einem Browserfeld

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.requestContent("http://www.blackberry.com");
    }
}
```

Anzeigen von HTML-Inhalten aus einer Ressource in Ihrer Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der mainScreen-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der BrowserField-Klasse.

```
BrowserField myBrowserField = new BrowserField();
```

5. Rufen Sie add() im Bildschirmkonstruktor auf, um dem Bildschirm das BrowserField-Objekt hinzuzufügen.

```
add(myBrowserField);
```

6. Rufen Sie BrowserField.requestContent() im Bildschirmkonstruktor auf, um den Ort der Ressource in Ihrer Anwendung festzulegen und den HTML-Inhalt anzuzeigen.

```
myBrowserField.requestContent("local:///test.html");
```

Anmerkung: Die BrowserField-Klasse greift nicht mithilfe einer Ordnerstruktur auf Ressourcen zu. Die BrowserField-Klasse zeigt die erste gefundene Ressource an, die dem angegebenen Dateinamen entspricht.

Codebeispiel: Anzeigen des HTML-Inhalts von einer Ressource in Ihrer Anwendung

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
```

```
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField myBrowserField = new BrowserField();
        add(myBrowserField);
        myBrowserField.requestContent("local:///test.html");
    }
}
```

Konfigurieren eines Browserfeldes

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `BrowserFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der `BrowserFieldConfig`-Klasse.

```
BrowserFieldConfig myBrowserFieldConfig = new BrowserFieldConfig();
```

5. Rufen Sie im Bildschirmkonstruktor `BrowserFieldConfig.setProperty()` auf, um eine Eigenschaft von `BrowserField` festzulegen. Der erste Parameter in `setProperty()` legt die Eigenschaft und der zweite Parameter den Wert der Eigenschaft fest. Zum Beispiel legt das folgende Codebeispiel die `NAVIGATION_MODE`-Eigenschaft eines `BrowserField`-Objekts fest:

```
myBrowserFieldConfig.setProperty(BrowserFieldConfig.NAVIGATION_MODE,
    BrowserFieldConfig.NAVIGATION_MODE_POINTER);
```

6. Erstellen Sie im Bildschirmkonstruktor eine Instanz der BrowserField-Klasse, die die von Ihnen definierte Konfiguration verwendet.

```
BrowserField browserField = new BrowserField(myBrowserFieldConfig);
```

Codebeispiel: Konfigurieren eines Browserfelds

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserFieldConfig myBrowserFieldConfig = new BrowserFieldConfig();
        myBrowserFieldConfig.setProperty(BrowserFieldConfig
            .NAVIGATION_MODE, BrowserFieldConfig.NAVIGATION_MODE_POINTER);
        BrowserField browserField = new BrowserField(myBrowserFieldConfig);
    }
}
```

Senden von Formulardaten an eine Webadresse in einem Browserfeld

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.io.http.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der UiApplication-Klasse. Erstellen Sie unter main() eine Instanz der neuen Klasse, und rufen Sie enterEventDispatcher() auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie pushScreen() im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene BrowserFieldDemoScreen-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der MainScreen-Klasse.

```
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Instanz der BrowserField-Klasse.

```
BrowserField browserField = new BrowserField();
```

5. Rufen Sie add() im Bildschirmkonstruktor auf, um dem Bildschirm das BrowserField-Objekt hinzuzufügen.

```
add(browserField);
```

6. Erstellen Sie im Bildschirmkonstruktor ein String-Objekt, das die Basis-Webadresse der Webseite enthält, an die Sie die Formulardaten senden.

```
String baseUrl = "http://www.blackberry.com";
```

7. Erstellen Sie im Bildschirmkonstruktor ein String-Objekt, das die Formulardaten angibt, die Ihre Anwendung an die Webseite sendet.

```
String postData = "fieldname1=value1&fieldname2=value2";
```

8. Erstellen Sie im Bildschirmkonstruktor ein Hashtable-Objekt, um die Kopfzeileninformationen für die Formulardaten zu speichern.

```
Hashtable header = new Hashtable();
```

9. Rufen Sie im Bildschirmkonstruktor Hashtable.put() auf, um die Kopfzeileninformationen für die Formulardaten anzugeben.

```
header.put("Content-Length", "" + postData.length());
header.put("Content-Type", "application/x-www-form-urlencoded");
```

10. Rufen Sie im Bildschirmkonstruktor BrowserField.requestContent() auf, um die Formulardaten an die Webseite zu senden und die Webseite anzuzeigen.

```
browserField.requestContent(baseUrl, postData.getBytes(), new
    HttpHeaders(header));
```

Codebeispiel: Senden von Formulardaten an eine Webadresse in einem Browserfeld

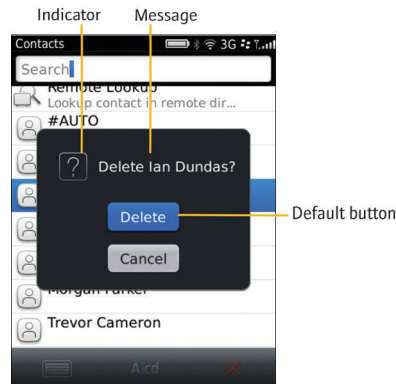
```
import net.rim.device.api.browser.field2.*;
import net.rim.device.api.io.http.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
import java.util.*;
public class BrowserFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        BrowserFieldDemo app = new BrowserFieldDemo();
        app.enterEventDispatcher();
    }
    public BrowserFieldDemo()
    {
        pushScreen(new BrowserFieldDemoScreen());
    }
}
class BrowserFieldDemoScreen extends MainScreen
{
    public BrowserFieldDemoScreen()
    {
        BrowserField browserField = new BrowserField();
        add(browserField);
        String baseUrl = "http://www.blackberry.com";
        String postData = "fieldname1=value1&fieldname2=value2";
        Hashtable header = new Hashtable();
        header.put("Content-Length", "" + postData.length());
        header.put("Content-Type", "application/x-www-form-urlencoded");
        browserField.requestContent(baseUrl, postData.getBytes(), new
        HttpHeaders(header));
    }
}
```

Dialogfelder

Verwenden Sie Dialogfelder, um die folgenden Aktionen auszuführen:

- Fordern Sie von den Benutzern Informationen, die erforderlich sind, um eine vom Benutzer eingeleitete Aufgabe abzuschließen.
- Teilen Sie den Benutzern dringende Informationen mit oder informieren Sie sie über den Status wichtiger Aktionen.
- Geben Sie an die Benutzer Warnmeldungen zu unerwarteten oder möglicherweise zerstörenden Bedingungen/Situationen aus.

Dialogfelder sind modal; sie unterbrechen den normalen Betrieb des BlackBerry®-Geräts und werden im Stapel an den Anfang geschoben. Ein Dialogfeld enthält eine Nachricht, Schaltflächen, anhand derer die Benutzer eine Aktion ausführen können, und eine Anzeige, die die Art des Dialogfelds angibt. Die Größe des Dialogfelds hängt von der Größe des BlackBerry-Gerätebildschirms ab. Die vom Benutzer auf dem BlackBerry ausgewählte Skin bestimmt den Anzeigestil des Dialogfelds.



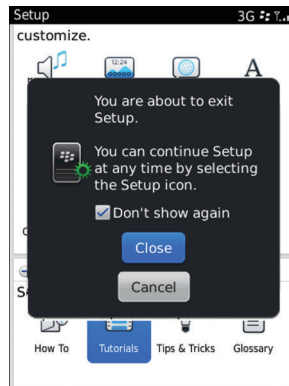
Bewährtes Verfahren: Implementieren von Dialogfeldern

- Verwenden Sie Schaltflächen zum Bestätigen oder Abbrechen von Aktionen in Dialogfeldern. Vermeiden Sie die Verwendung von Links oder anderen Komponenten.
- Verwenden Sie eine Standardanzeige, die der Art des Dialogfelds entspricht. Vermeiden Sie die Verwendung von Mehrfachanzeigen in einem Dialogfeld.
- Vermeiden Sie wenn möglich, dass Benutzer in einem Dialogfeld scrollen müssen. Fügen Sie Bildlaufpfeile hinzu, wenn Ihre Dialogfeldnachricht oder Schaltflächen im Dialogfeld nicht vollständig angezeigt werden können. Wenn Sie Standardkomponenten verwenden, werden Bildlaufpfeile bei Bedarf automatisch angezeigt.
- Geben Sie Benutzern immer die Möglichkeit, ein Dialogfeld mit der Esc-Taste zu schließen. Vermeiden Sie das Implementieren einer anderen Aktion, wenn Benutzer zum Schließen eines Dialogfelds die Esc-Taste drücken. Wenn ein Dialogfeld Benutzern beispielsweise die Möglichkeit gibt, eine Einstellung zu ändern, implementieren Sie keine Änderungen, wenn Benutzer die Esc-Taste drücken. Zeigen Sie bei Bedarf das Dialogfeld zu einem späteren Zeitpunkt an.
- Wenn ein Dialogfeld auf einem Anwendungsbildschirm angezeigt wird und Benutzer den Ein-/Aus-Schalter drücken, zeigen Sie die Startseite oder die Anwendungsliste an. Wenn Benutzer zur Anwendung zurückkehren, zeigen Sie wieder das Dialogfeld an.

Richtlinien für das Layout

- Zentrieren Sie das Dialogfeld auf dem Bildschirm. Wenn Sie Standardkomponenten verwenden, zentriert das BlackBerry®-Gerät automatisch das Dialogfeld.
- Erstellen Sie Dialogfelder, die maximal 90 % der Bildschirmbreite und -höhe einnehmen. Wenn Sie Standardkomponenten verwenden, berechnet das BlackBerry-Gerät automatisch die entsprechende Größe der Dialogfelder.
- Zentrieren Sie die Dialogfeldanzeige senkrecht zur Dialogfeldnachricht.

- Zeigen Sie Meldungen rechts von der Anzeige und oberhalb der Schaltflächen (für die meisten Sprachen) an.
- Platzieren Sie Schaltflächen für Bestätigungsaktionen zuerst. Platzieren Sie beispielsweise "Speichern" vor "Verwerfen" oder "Abbrechen".
- Zentrieren Sie Schaltflächen in Dialogfeldern horizontal.
- Platzieren Sie Schaltflächen senkrecht in dem Dialogfeld. Das senkrechte Layout ermöglicht die Erweiterung von Schaltflächen, um lokalisierte Schaltflächenbezeichnungen anzupassen.
- Wenn Sie ein Kontrollkästchen verwenden, passen Sie die Ausrichtung des Kontrollkästchens an die Ausrichtung der Dialogfeldmeldung an. Platzieren Sie das Kontrollkästchen über den Schaltflächen. Das Kontrollkästchen sollte standardmäßig aktiviert sein, es sei denn, das Dialogfeld zeigt eine Meldung mit kritischen Informationen für Benutzer an.



Richtlinien für Nachrichten

- Seien Sie möglichst genau. Falls möglich, verwenden Sie einen kurzen Satz, der deutlich den Grund zum Anzeigen des Dialogfelds und der Aktionen erklärt, die es schließen können.
- Verwenden Sie nach Möglichkeit für Nachrichten vollständige Sätze.
- Verwenden Sie den Wortschatz, den Benutzer verstehen. Verwenden Sie zum Beispiel "Die Datei konnte nicht gespeichert werden, weil die Medienkarte voll ist" statt "Fehler beim Schreiben der Datei auf den Datenträger".
- Verwenden Sie wo möglich positive Sprache, um ein Tadeln der Benutzer zu vermeiden. Schreiben Sie nie Nachrichten, die Benutzer für Fehler oder unerwartete Bedingungen tadeln. Konzentrieren Sie sich stattdessen auf die Aktionen, die Benutzer unternehmen können, um das Problem zu lösen.
- Verwenden Sie bei der Ansprache der Benutzer die zweite Person (Sie, Ihr).
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Vermeiden Sie in Nachrichten die Verwendung von Ausrufezeichen (!).
- Vermeiden Sie die Verwendung von Auslassungspunkten (...) in Nachrichten, es sei denn, Sie zeigen einen Fortschritt an (zum Beispiel, "Bitte warten...").

Richtlinien für Schaltflächen

- Verwenden Sie für die Standardschaltfläche jene Schaltfläche, die Benutzer am wahrscheinlichsten anklicken. Vermeiden Sie die Verwendung einer Schaltfläche, die mit einer beendenden Aktion verknüpft ist, als Standardschaltfläche. Ausnahmen zu dieser Regel sind die Fälle, bei denen Benutzer eine Aktion auslösen, die nur eine geringfügig destruktive Auswirkung haben (wie z. B. das Löschen eines einzelnen Elements) und die häufigste Benutzeraktion darin besteht, mit der Aktion fortzufahren.
- Vermeiden Sie die Verwendung von mehr als drei Schaltflächen in einem Dialogfeld. Wenn Sie mehr als drei benötigen, sollten Sie statt den Optionsfeldern einen Anwendungsbildschirm verwenden.
- Stellen Sie auf BlackBerry-Geräten mit einer physischen Tastatur Tastenkombinationen für Schaltflächen bereit. In der Regel ist die Tastenkombination der erste Buchstabe der Schaltflächenbezeichnung.
- Verwenden Sie kurze, klare Bezeichnungen.
- Verwenden Sie nach Möglichkeit Bezeichnungen, die aus einem Wort bestehen. Die Größe einer Schaltfläche ändert sich mit der Länge der Bezeichnung. Wenn eine Bezeichnung zu lang ist, zeigt ein Auslassungspunkt (...) an, dass der Text abgeschnitten ist.
- Vermeiden Sie die Verwendung der Bezeichnungen "Ja" und "Nr." Verwenden Sie für Bezeichnungen Verben, die die verknüpfte Aktion beschreiben (zum Beispiel "Abbrechen", "Löschen", "Verwerfen" oder "Speichern"). Dieser Ansatz hilft Benutzern schnell und problemlos zu verstehen, was geschieht, wenn sie auf die Schaltfläche klicken. Falls erforderlich, fügen Sie weitere Beschreibungen anderswo auf dem Bildschirm hinzu (zum Beispiel in einer Anwendungsnachricht).
- Vermeiden Sie die Verwendung von Symbolen oder Grafiken in Bezeichnungen.
- Vermeiden Sie Interpunktion in Bezeichnungen. Verwenden Sie in der Bezeichnung einer Schaltfläche Auslassungspunkte um anzuzeigen, dass zusätzliche Informationen erforderlich sind, bevor die verknüpfte Aktion ausgeführt werden kann.

Erstellen eines Dialogfelds

Verwenden Sie Alarmdialogfelder, um Benutzer auf eine kritische Aktion hinzuweisen, wie z. B. das Ausschalten des BlackBerry®-Geräts, oder auf einen Fehler, wie z. B. das Eingeben ungültiger Informationen. In einem Alarmdialogfeld wird ein Ausrufezeichen (!) angezeigt. Um ein Alarmdialogfeld zu schließen, können Benutzer auf "OK" klicken oder die Esc-Taste drücken. Weitere Informationen zu anderen Arten von Dialogfeldern finden Sie in der API-Referenz für die BlackBerry® Java® Development Environment.

1. Importieren Sie die `net.rim.device.api.ui.component.Dialog`-Klasse.
2. Erstellen Sie ein Alarmdialogfeld, das den Alarmtext enthält, den Sie anzeigen möchten.

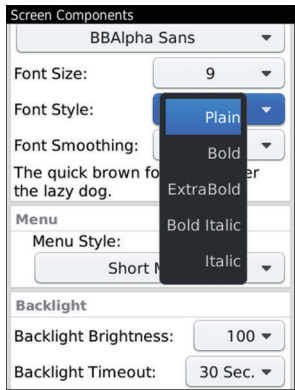
```
Dialog.alert("Specify the alert text that you want to display.")
```

Dropdown-Listen

Verwenden Sie Dropdown-Listen, um einen Satz sich gegenseitig ausschließender Werte bereitzustellen.

Benutzer können mit einer Dropdown-Liste die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Klicken auf einen Wert in einer Dropdown-Liste	Drücken der Eingabetaste oder Klicken mit dem Trackpad	• Tippen auf das Element • Drücken Sie die Eingabetaste. • Klicken mit dem Trackpad.



Bewährtes Verfahren: Implementieren von Dropdown-Listen

- Verwenden Sie eine Dropdown-Liste, um zwei oder mehr Auswahlmöglichkeiten anzubieten, wenn der Platz beschränkt ist. Wenn der Platz kein Problem ist, sollten Sie stattdessen Optionsfelder verwenden, sodass die Benutzer die verfügbaren Optionen auf dem Bildschirm anzeigen können.
- Verwenden Sie zur Erstellung von Dropdown-Listen die `ObjectChoiceField`- oder `NumericChoiceField`-Klasse.
- Verwenden Sie für den Standardwert jenen Wert, den Benutzer am wahrscheinlichsten anklicken.
- Verwenden Sie die markierte Option als Standardfokus, wenn Benutzer durch die Liste scrollen.
- Wenn Benutzer nicht auf einen Wert klicken müssen, nehmen Sie einen Wert wie "Kein/e/s" in die Dropdown-Liste auf. Platzieren Sie den Wert "Kein/e/s" immer ganz oben in der Liste.
- Gruppieren und ordnen Sie die Werte möglichst logisch (fassen Sie zum Beispiel zusammenhängende Werte in Gruppen zusammen oder beginnen Sie mit den gebräuchlichsten Werten). Vermeiden Sie es, die Werte alphabetisch anzuordnen, weil die alphabetische Reihenfolge sprachspezifisch ist.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen für Dropdown-Listen und für die Werte in einer Dropdown-Liste. Vergewissern Sie sich, dass die Bezeichnung klar beschreibt, was geschieht, wenn Benutzer auf den Wert klicken. Die Breite der Dropdown-Liste ändert sich basierend auf der Länge der Wertbezeichnungen. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist.
- Vermeiden Sie die Bezeichnungen "Ja" und "Nein" als Dropdown-Listenwerte. Formulieren Sie die Option um und verwenden Sie stattdessen ein Kontrollkästchen.

- Platzieren Sie die Bezeichnung auf der linken Seite der Dropdown-Liste.
- Verwenden Sie die Groß-/Kleinschreibung, um die Dropdown-Listenbezeichnungen und Werte zu betiteln (es sei denn, Sie bezeichnen die Werte mit einem Satz).
- Schließen Sie die Bezeichnungen von Dropdown-Listen mit einem Doppelpunkt (:) ab.

Erstellen einer Dropdown-Liste

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor mithilfe der `ObjectChoiceField`-Klasse eine Dropdown-Liste, die eine Liste von Wörtern und Formulierungen anzeigt. Erstellen Sie ein `String`-Array, um die Elemente zu speichern, die Sie in der Dropdown-Liste anzeigen möchten. Erstellen Sie ein `int`-Objekt, in dem das Standardelement gespeichert wird, das in der Dropdown-Liste angezeigt werden soll. Geben Sie im `ObjectChoiceField`-Konstruktor die Bezeichnung für die Dropdown-Liste, das Array der anzuzeigenden Elemente und das Standardelement an. Im folgenden Codebeispiel wird standardmäßig "Wednesday" (Mittwoch) angezeigt. Rufen Sie `add()` auf, um die Dropdown-Liste dem Bildschirm hinzuzufügen.

```
String choices[] =
{"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
int iSetTo = 2;
add(new ObjectChoiceField("First Drop-down List", choices, iSetTo));
```

- Erstellen Sie im Bildschirmkonstruktor mithilfe der `NumericChoiceField`-Klasse eine zweite Dropdown-Liste, die eine Liste von Zahlen anzeigt. Geben Sie im `NumericChoiceField`-Konstruktor die Bezeichnung für die Dropdown-Liste, die erste und die letzte in der Dropdown-Liste anzuzeigende Zahl, die für die Zahlenliste zu verwendende Schrittgröße und die Standardzahl an. Im folgenden Codebeispiel werden die numerischen Parameter in `int`-Objekten gespeichert. In der Dropdown-Liste sind die Zahlen 1 bis 31 enthalten und standardmäßig wird die Zahl 10 angezeigt. Rufen Sie `add()` auf, um die zweite Dropdown-Liste dem Bildschirm hinzuzufügen.

```
int iStartAt = 1;
int iEndAt = 31;
int iIncrement = 1;
iSetTo = 10;
add(new NumericChoiceField("Numeric Drop-Down
List", iStartAt, iEndAt, iIncrement, iSetTo));
```

- Um die Standardfunktionalität zu überschreiben, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, überschreiben Sie in der Erweiterung der `MainScreen`-Klasse die Methode `MainScreen.onSavePrompt()`. Im folgenden Codebeispiel wird `true` zurückgegeben. Dies bedeutet, dass vor dem Schließen keine Benutzeraufforderung durch die Anwendung erfolgt.

```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen einer Dropdown-Liste

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        String choices[] =
```

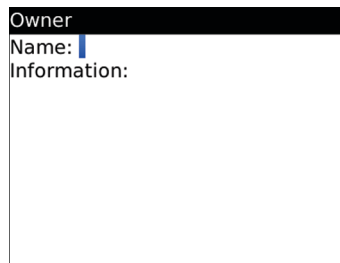
```

{"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
    int iSetTo = 2;
    add(new ObjectChoiceField("First Drop-down List", choices, iSetTo));
    int iStartAt = 1;
    int iEndAt = 31;
    int iIncrement = 1;
    iSetTo = 10;
    add(new NumericChoiceField("Numeric Drop-Down
List", iStartAt, iEndAt, iIncrement, iSetTo));
}
public boolean onSavePrompt()
{
    return true;
}
}

```

Bezeichnungen

Verwenden Sie eine Bezeichnung, um Text anzuzeigen, der eine Steuerung kennzeichnet.



Bewährtes Verfahren: Implementieren von Bezeichnungen

- Verwenden Sie die `LabelField`-Klasse, um Bezeichnungen zu erstellen.
- Verwenden Sie kurze, klare Bezeichnungen.
- Gruppieren und ordnen Sie Bezeichnungen logisch an (gruppieren Sie beispielsweise zusammen gehörende Elemente oder führen Sie die häufigsten Elemente zuerst an). Vermeiden Sie es, die Werte alphabetisch anzuordnen, weil die alphabetische Reihenfolge sprachspezifisch ist.
- Interpunktieren Sie die Bezeichnung mit einem Doppelpunkt (:).

Erstellen eines Textetiketts

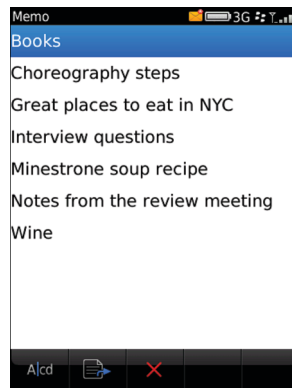
1. Importieren Sie die `net.rim.device.api.ui.component.LabelField`-Klasse.
2. Erstellen Sie eine Instanz von `LabelField`, um einem Bildschirm ein Textetikett hinzuzufügen.

```
LabelField title = new LabelField("UI Component Sample", LabelField.ELLIPSIS);
```

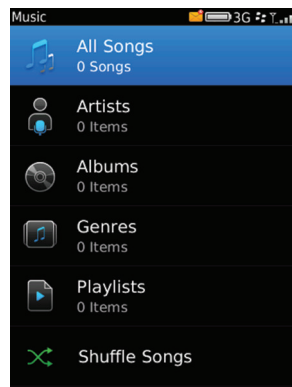
Listen und Tabellen

Verwenden Sie Listen und Tabellen, um die Elemente anzuzeigen, die Benutzer markieren und öffnen können. Wenn die Liste lang ist, werden die Elemente in Stapeln abgerufen und angezeigt. Wenn Benutzer das letzte Element in der Liste erreichen, wird der nächste Elementenstapel am Ende der Liste angezeigt.

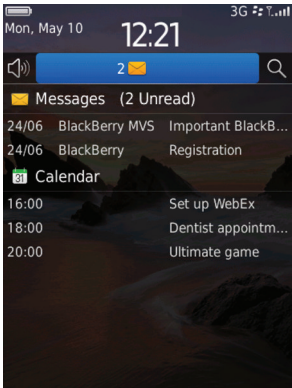
Verwenden Sie eine einfache Liste, um Text problemlos in Zeilen anzuzeigen.



Verwenden Sie eine erweiterte Liste, um Textzeilen und Symbole anzuzeigen. Derzeit zeigen erweiterte Listen nur Informationen an und sind nicht interaktiv.



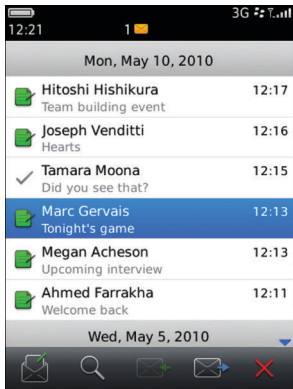
Wenn Sie Elemente in Spalten und Zeilen anzeigen möchten, verwenden Sie eine Tabelle.



Sie können Elemente unter Kopfzeilen anordnen, damit die Benutzer einfacher durch lange Listen navigieren können. Zum Beispiel können Sie Kopfzeilen erstellen, die ausgeblendet werden, wodurch die Benutzer die Elemente in der Liste einfacher finden können.

Demo		
3.6		
	5810	
	2002	keyboard
	6210	
	2002	keyboard
	6220	
	2002	keyboard
	6710	
	2002	keyboard
3.7		
	7210	
	2003	keyboard
4.0		
	7100g	
	2004	keyboard
	7510	
	2004	keyboard
	7730	
	2004	keyboard

Sie können auch Elemente unter Kopfzeilen anordnen, die immer am Anfang der Liste angezeigt werden. Zum Beispiel können Sie ein Datum als Kopfzeile hinzufügen und unter der Kopfzeile die Nachrichten anordnen, die an diesem Datum empfangen wurden. Benutzer können eine Kopfzeile markieren, um eine Aktion für die Gruppe der Elemente auszuführen, oder sie können Tastenkombinationen verwenden, um durch die Liste zu navigieren.



Benutzer können die folgenden Aktionen in Listen und Tabellen ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Durch die Elemente der Liste scrollen.	Fahren Sie mit einem Finger senkrecht über das Trackpad.	• Ziehen Sie einen Finger senkrecht über den Bildschirm. • Führen Sie den Finger auf dem Bildschirm nach oben oder nach unten. • Fahren Sie mit einem Finger senkrecht über das Trackpad.
Ein Element in der Liste markieren.	Fahren Sie mit einem Finger senkrecht über das Trackpad.	• Berühren Sie das Element leicht. • Fahren Sie mit einem Finger senkrecht über das Trackpad.
Ein Element in der Liste öffnen.	• Klicken mit dem Trackpad. • Drücken Sie die Eingabetaste.	• Tippen auf das Element • Klicken mit dem Trackpad. • Drücken Sie die Eingabetaste.

Bewährtes Verfahren: Implementieren von Listen und Tabellen

- Verwenden Sie die SimpleList-Klasse, um eine Liste mit Text zu erstellen. Verwenden Sie die RichList-Klasse, um eine schreibgeschützte Liste mit Text und Symbolen zu erstellen. Verwenden Sie die TableView-Klasse, um eine interaktive erweiterte Liste mit Text und Symbolen zu erstellen.
- Verwenden Sie die TableView-Klasse, um eine Tabelle mit Zeilen und Spalten zu erstellen. Sie können die GridFieldManager-Klasse verwenden, wenn Sie mit einer festen Anzahl von Zeilen und Spalten arbeiten.
- Wenn die Liste lang ist, und Sie die Elemente auf separaten Bildschirmen anzeigen möchten, stellen Sie unten auf dem Bildschirm die Schaltflächen "Weiter" und "Zurück" bereit. Wenn die Liste sehr lang ist (z. B. Tausende von Elementen), sollten Sie stattdessen Bildschirmnummern bereitstellen.

- Wenn Sie erwarten, dass die Benutzer in den Elementen der Liste blättern (zum Beispiel in einer Nachrichtenliste oder einem Feed), weisen Sie Tastenkombinationen zu, um zum nächsten oder vorherigen Listenelement zu gelangen. Wenn möglich, erlauben Sie es den Benutzern der deutschen Version, mit "W" zum nächsten Listenelement und mit "Z" zum vorherigen Listenelement zu gelangen.

Erstellen eines Listenfelds

Verwenden Sie ein Listenfeld, um eine Liste anzuzeigen, aus der Benutzer eine oder mehrere Werte auswählen können.

1. Importieren Sie die folgenden Klassen:

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.util.Vector;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `CreateMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar, der in Schritt 3 beschrieben wird.

```
public class ListFields extends UiApplication
{
    public static void main(String[] args)
    {
        ListFields theApp = new ListFields();
        theApp.enterEventDispatcher();
    }
    public ListFields()
    {
        pushScreen(new ListFieldScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Konstruktor `setTitle()` auf, um den Titel für den Bildschirm anzuzeigen. Rufen Sie `add()` auf, um ein Textfeld auf dem Bildschirm anzuzeigen. Rufen Sie `addMenuItem()` auf, um einen Menüeintrag zu dem Menü hinzuzufügen, das `MainScreen` erstellt.

```
class ListFieldScreen extends MainScreen
{
    private ListField _listField;
    private Vector _listElements;
    public ListFieldScreen()
    {
        setTitle("List Field Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor das Listenfeld. Erstellen Sie ein Array für die Elemente, die Sie dem Listenfeld mithilfe der Vector-Klasse hinzufügen möchten. Erstellen Sie das Listenfeld mithilfe der ListField() Klasse. Rufen Sie add() auf, um dem Bildschirm das Listenfeld hinzuzufügen. Rufen Sie initializeList() auf, die in Schritt 4 beschrieben ist, um ein Listenfeld hinzuzufügen.

```
_listElements = new Vector();
_listField = new ListField();
ListCallback _callback = new ListCallback();
_listField.setCallback(_callback);
add(_listField);
initializeList();
```

5. Erstellen Sie eine Methode, um die Elemente anzugeben, die Sie im Listenfeld mithilfe des String-Objekts anzeigen wollen. Rufen Sie addElement() auf, um die Elemente der Liste hinzuzufügen. Rufen Sie setSize() auf, um die Anzahl der Elemente im Listenfeld anzugeben.

```
private void initializeList()
{
    String itemOne = "List item one";
    String itemTwo = "List item two";
    _listElements.addElement(itemOne);
    _listElements.addElement(itemTwo);
    reloadList();
}
private void reloadList()
{
    _listField.setSize(_listElements.size());
}
```

6. Erstellen Sie eine Klasse, die die ListFieldCallback-Schnittstelle implementiert. Implementieren Sie drawListRow(), um die Listenfeldelemente zum Bildschirm hinzuzufügen. Implementieren Sie get(), um das Listenfeldelement an den angegebenen Index zurückzugeben. Implementieren Sie indexOfList(), um das erste Vorkommen einer gegebenen Zeichenfolge zurückzugeben. Implementieren Sie getPreferredWidth(), um die Breite des Listenfelds abzurufen.

```
private class ListCallback implements ListFieldCallback
{
    public void drawListRow(ListField list, Graphics g, int index, int y, int w)
    {
        String text = (String)_listElements.elementAt(index);
        g.drawText(text, 0, y, 0, w);
    }
    public Object get(ListField list, int index)
    {
        return _listElements.elementAt(index);
    }
    public int indexOfList(ListField list, String prefix, int string)
    {
        return _listElements.indexOf(prefix, string);
    }
    public int getPreferredWidth(ListField list)
    {

```

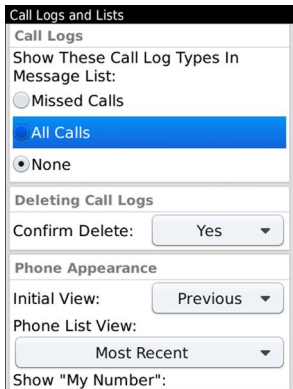
```
        return Display.getWidth();
    }
}
```

Optionsfelder

Verwenden Sie Optionsfelder, um einen Satz sich gegenseitig ausschließender aber verwandter Auswahlen anzuzeigen.

Benutzer können die folgende Aktion mit Optionsfeldern ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Auswählen eines Optionsfelds.	Drücken der Leertaste oder klicken mit dem Trackpad	• Tippen auf das Element • Drücken Sie die Leertaste. • Klicken mit dem Trackpad.



Bewährtes Verfahren: Implementieren von Optionsfeldern

- Verwenden Sie Optionsfelder, um zwei oder mehr Auswahlmöglichkeiten anzubieten, wenn der Platz nicht beschränkt ist. Wenn der Platz beschränkt ist, erwägen Sie, stattdessen eine Dropdown-Liste zu verwenden.
- Verwenden Sie die `RadioButtonField`-Klasse, um Optionsfelder zu erstellen.
- Vergewissern Sie sich, dass der Inhalt für Optionsfelder statisch bleibt. Der Inhalt sollte sich für Optionsfelder nicht kontextbezogen ändern.
- Starten Sie keine Aktion, wenn Benutzer ein Optionsfeld aktivieren. Öffnen Sie zum Beispiel keinen neuen Bildschirm.
- Richten Sie Optionsfelder senkrecht aus.

- Gruppieren und ordnen Sie die Werte möglichst logisch (fassen Sie zum Beispiel zusammenhängende Optionsfelder in Gruppen zusammen oder beginnen Sie mit den gebräuchlichsten Werten). Vermeiden Sie es, die Optionsfelder alphabetisch anzuordnen, da die alphabetische Reihenfolge sprachspezifisch ist.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare Bezeichnungen. Stellen Sie sicher, dass die Bezeichnung klar beschreibt, was ausgeführt wird, wenn Benutzer das Optionsfeld aktivieren. Wenn die Bezeichnungen zu lang sind, erfolgt ein Zeilenumbruch.
- Platzieren Sie Bezeichnungen rechts von den Optionsfeldern.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Verwenden Sie keine Satzzeichen am Ende der Bezeichnung.

Erstellen eines Optionsfeldes

Mithilfe der `RadioButtonGroup`-Klasse können Sie eine Gruppe von Optionsfeldern erstellen. Ein Benutzer kann nur eine Option aus einer Gruppe von Optionsfeldern auswählen.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUIScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUIScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUIScreen extends MainScreen
{
    public MyUIScreen()
    {
```

```
        setTitle("UI Component Sample");
    }
}
```

4. Erstellen Sie im Bildschirmkonstruktor eine Gruppe von Optionsfeldern und verwenden Sie dazu die `RadioButtonGroup`-Klasse. Erstellen Sie die Optionsfelder, die Sie der Gruppe mithilfe der `RadioButtonField`-Klasse hinzufügen möchten. Geben Sie im `RadioButtonField`-Konstruktor die Bezeichnung für das Optionsfeld, die Gruppe und einen booleschen Wert an, um die Standardauswahl zu definieren (zum Beispiel bedeutet `true`, dass standardmäßig die Option ausgewählt ist). Rufen Sie `add()` auf, um dem Bildschirm die Optionsfelder hinzuzufügen.

```
RadioButtonGroup rbg = new RadioButtonGroup();
add(new RadioButtonField("Option 1",rbg,true));
add(new RadioButtonField("Option 2",rbg,false));
```

5. Um die Standardfunktionalität zu überschreiben, die den Benutzer auffordert, Änderungen vor dem Schließen der Anwendung zu speichern, überschreiben Sie in der Erweiterung der `MainScreen`-Klasse die Methode `MainScreen.onSavePrompt()`. Im folgenden Codebeispiel wird `true` zurückgegeben. Dies bedeutet, dass vor dem Schließen keine Benutzeraufforderung durch die Anwendung erfolgt.

```
public boolean onSavePrompt()
{
    return true;
}
```

Codebeispiel: Erstellen eines Optionsfeldes

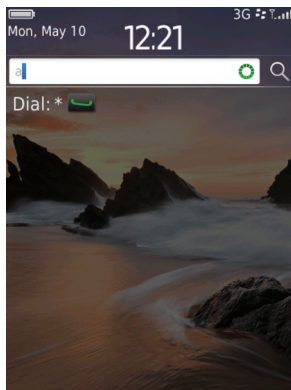
```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
        RadioButtonGroup rbg = new RadioButtonGroup();
        add(new RadioButtonField("Option 1",rbg,true));
        add(new RadioButtonField("Option 2",rbg,false));
    }
    public boolean onSavePrompt()
    {
```

```
        return true;  
    }  
}
```

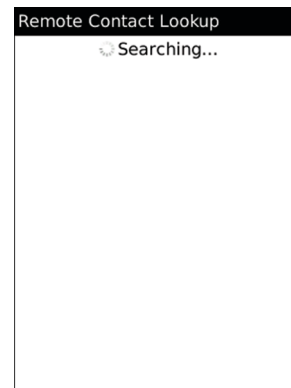
Aktivitäts- und Fortschrittsanzeigen

Aktivitäts- und Fortschrittsanzeigen zeigen den Benutzern, dass ihre BlackBerry®-Geräte eine Aktion ausführen, z. B. das Suchen nach Elementen oder das Entfernen von Sprachen.

Verwenden Sie eine Aktivitätsanzeige, wenn Sie zeigen möchten, dass das BlackBerry®-Gerät aktiv ist und Sie nicht bestimmen können, wie lange die Aktion dauert. Sie können jeder beliebigen Komponente eine Aktivitätsanzeige, z. B. einen Bildschirm, ein Textfeld oder einen Listeneintrag, hinzufügen. Außerdem können Sie einer Aktivitätsanzeige Text hinzufügen, um die Aktion zu beschreiben.

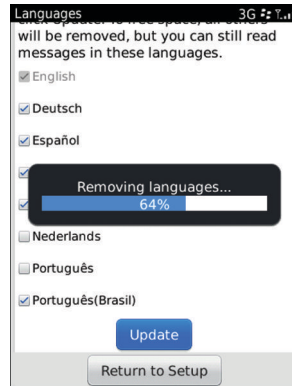


Eine Aktivitätsanzeige in einem Feld



Eine Aktivitätsanzeige mit Text

Verwenden Sie eine Fortschrittsanzeige, wenn Sie die Dauer einer Aktion bestimmen können. Fortschrittsanzeigen enthalten neben einer Bezeichnung, die die Aktion beschreibt, eine horizontale Leiste, die sich mit dem Aktionsfortschritt von links nach rechts füllt. Ein Prozentsatz zeigt in der Leiste an, wie viel der Aktion bereits verarbeitet wurde.



Bewährtes Verfahren: Implementieren von Aktivitäts- und Fortschrittsanzeigen

- Zeigen Sie immer den Fortschritt an, wenn für den Abschluss eines Vorgangs mehr als 2 Sekunden benötigt werden.
- Verwenden Sie eine Fortschrittsanzeige, wenn Sie die Dauer einer Aktion bestimmen können.
- Verwenden Sie eine Aktivitätsanzeige, wenn Sie die Dauer einer Aktion nicht bestimmen können.
- Verwenden Sie die `ActivityIndicatorView`-Klasse, um Aktivitätsanzeigen zu erstellen. Verwenden Sie die `ProgressIndicatorView`-Klasse, um Fortschrittsanzeigen zu erstellen.
- Stellen Sie hilfreiche Fortschrittsinformationen bereit. Wenn zum Beispiel Benutzer eine Anwendung auf ihr Gerät herunterladen, zeigen Sie den Prozentsatz der Daten an, den das BlackBerry®-Gerät heruntergeladen hat. Seien Sie bei den Fortschrittsinformationen so genau wie möglich.
- Ermöglichen Sie, dass die Benutzer immer die Beenden-Taste verwenden können, um eine Fortschrittsanzeige auszublenden.

Richtlinien für Text

- Verwenden Sie einen kurzen und aussagekräftigen Text (zum Beispiel "Daten werden geladen..." oder "Anwendungsliste wird erstellt...").
- Wenn eine Aktion lang andauert und Sie mitteilen möchten, was in jeder Phase geschieht, zeigen Sie einen Text an, der jede einzelne Phase beschreibt (zum Beispiel "Download wird ausgeführt..." oder "Installation wird ausgeführt...").
- Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.
- Interpunktieren Sie den Text mit Auslassungspunkten (...).

Anzeigen von Aktivität oder Aufgabenfortschritt

Sie können das im `net.rim.device.api.ui.component.progressindicator`-Paket bereitgestellte API für die Aktivitäts- und Fortschrittsanzeige verwenden, um anhand von visuellen Hinweisen auf dem Bildschirm anzuzeigen, dass eine Arbeit erledigt oder eine Aufgabe ausgeführt wird. Sie können nicht nur eine Aktivität darstellen, deren Zeitdauer unbekannt ist, sondern auch den Fortschritt, der numerisch ausgegeben werden kann (zum Beispiel als Prozentsatz einer abgeschlossenen Aufgabe).

Das API für die Aktivitäts- und Fortschrittsanzeige verwendet das Designmuster des Modell-Ansicht-Controllers und enthält zwei Felder, die für die Ausgabe der Aktivität oder des Fortschritts verantwortlich sind:

- Die `ActivityImageField`-Klasse stellt Aktivität mithilfe einer Bitmap dar, die Animations-Frames enthält. Der angezeigte Frame ändert sich im Zeitablauf. In der Regel handelt es sich bei diesem Feld um ein Drehfeld, eine Sanduhr oder um einen ähnlich animierten visuellen Hinweis. Dieses Feld wird mithilfe der `ActivityIndicatorView`-Klasse erstellt und festgelegt.
- Die `ProgressBarField`-Klasse stellt den Aufgabenfortschritt als Leiste dar, die sich während der Aufgabenausführung füllt, bis die Aufgabe abgeschlossen ist. Dieses Feld wird mithilfe der `ProgressIndicatorView`-Klasse erstellt und festgelegt.

Die Progress Indicator Demo-Beispielanwendung wird im BlackBerry® Java® SDK bereitgestellt. Diese Beispielanwendung veranschaulicht, wie verschiedene Aktivitätsanzeigen und eine Fortschrittsanzeigenleiste erstellt und geändert werden können.

Anzeigen der Aktivität

Sie können ein Feld in Ihrer BlackBerry®-Geräteanwendung anzeigen, das darauf hinweist, dass eine Aktivität ausgeführt wird. In der Regel handelt es sich bei diesem Feld um ein Drehfeld, eine Sanduhr oder um einen ähnlich animierten visuellen Hinweis. Das Feld wird mithilfe der `ActivityImageField`-Klasse implementiert.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.Bitmap;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.progressindicator.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie ein Bild, das Animations-Frames enthält, und fügen Sie es Ihrem Projekt hinzu. Das Bild muss aus einer Reihe von horizontal angeordneten Animations-Frames bestehen. Die Bildbreite muss der Breite eines Frames multipliziert mit die Anzahl der Frames entsprechen.

Zum Beispiel verfügt das folgende Bild über fünf gleich große Frames.



3. Erstellen Sie aus dem Bild eine Bitmap.

```
Bitmap bitmap = Bitmap.getBitmapResource("spinner.png");
```

4. Erstellen Sie ein `ActivityIndicatorView`-Objekt. Sie können einen Stil für die Ansicht im Konstruktor festlegen.

```
ActivityIndicatorView view = new ActivityIndicatorView(Field.USE_ALL_WIDTH);
```

Um einen Manager zur Verwendung von Layout und Fokus festzulegen, geben Sie beim Erstellen der `ActivityIndicatorView` ein zweites Argument an. Wenn Sie keinen Manager festlegen, wird ein `VerticalFieldManager`-Objekt verwendet.

- Erstellen Sie ein Modell und einen Controller.

```
ActivityIndicatorModel model = new ActivityIndicatorModel();
ActivityIndicatorController controller = new ActivityIndicatorController();
```

- Verbinden Sie die Ansicht, das Modell und den Controller.

```
view.setController(controller);
view.setModel(model);
controller.setModel(model);
controller.setView(view);
model.setController(controller);
```

- Erstellen Sie das Feld, das die Aktivität aus dem Bitmap ausgibt.

```
view.createActivityImageField(bitmap, 5, Field.FIELD_HCENTER);
```

In diesem Beispiel besteht das Bitmap aus fünf Frames und wird im Manager der Ansicht zentriert angezeigt.

- Fügen Sie Ihrem Bildschirm die Ansicht hinzu.

```
add(view);
```

- Steuern Sie die Aktivitätsanzeige, indem Sie bei Bedarf das Modell der Ansicht steuern.

```
MenuItem _stopIndicator = new MenuItem("Stop spinner", 66000, 0)
{
    public void run()
    {
        view.getModel().cancel();
    }
};
MenuItem _resumeIndicator = new MenuItem("Resume spinner", 66010, 0)
{
    public void run()
    {
        view.getModel().resume();
    }
};
```

Dieses Beispiel ruft die `cancel()`-Methode des Modells auf, um die Animation anzuhalten. Das Beispiel ruft die `resume()`-Methode des Modells auf, um die Animation wieder aufzunehmen.

Codebeispiel

```
public class ActivityIndicatorScreen extends MainScreen
{
    ActivityIndicatorView view = new ActivityIndicatorView(Field.USE_ALL_WIDTH);
    ActivityIndicatorModel model = new ActivityIndicatorModel();
    ActivityIndicatorController controller = new ActivityIndicatorController();
    public ActivityIndicatorScreen ()
```

```

{
    setTitle("Activity Indicator Demo");
    view.setController(controller);
    view.setModel(model);
    controller.setModel(model);
    controller.setView(view);
    model.setController(controller);
    // Define the indicator image and create a field from it
    Bitmap bitmap = Bitmap.getBitmapResource("spinner.png");
    view.createActivityImageField(bitmap, 5, Field.FIELD_HCENTER);
    // add the view to the screen
    add(view);
    MenuItem _stopIndicator = new MenuItem("Stop spinner", 66000, 0)
    {
        public void run()
        {
            view.getModel().cancel();
        }
    };
    MenuItem _resumeIndicator = new MenuItem("Resume spinner", 66010, 0)
    {
        public void run()
        {
            view.getModel().resume();
        }
    };
    addMenuItem(_stopIndicator);
    addMenuItem(_resumeIndicator);
}
}

```

In der im BlackBerry® Java® SDK bereitgestellten Progress Indicator Demo-Beispielanwendung werden eine Reihe von Aktivitätsanzeigen, einschließlich des oben angezeigten Drehfelds, erstellt und geändert.

Anzeigen des Fortschritts

Sie können ein Feld in Ihrer BlackBerry®-Geräteanwendung anzeigen, das darauf hinweist, dass eine Aufgabe ausgeführt wird. Der Fortschritt wird durch eine Leiste dargestellt, die sich während der Aufgabenausführung füllt, bis die Aufgabe abgeschlossen ist.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.progressindicator.*;

```

2. Erstellen Sie ein ProgressIndicatorView-Objekt. Sie können einen Stil für die Ansicht im Konstruktor festlegen. Im folgenden Beispiel ist kein Stil festgelegt.

```

ProgressIndicatorView view = new ProgressIndicatorView(0);

```

Um einen Manager zur Verwendung von Layout und Fokus festzulegen, geben Sie beim Erstellen der ProgressIndicatorView ein zweites Argument an. Wenn Sie keinen Manager festlegen, wird ein VerticalFieldManager-Objekt verwendet.

- Erstellen Sie ein `ProgressIndicatorController`-Objekt.

```
ProgressIndicatorController controller = new ProgressIndicatorController();
```

- Erstellen Sie ein `ProgressIndicatorModel`-Objekt. Dieses Objekt stellt den Fortschritt der Aufgabe dar. Wenn Sie das Objekt erstellen, können Sie den Anfangswert des Modells, seinen Maximalwert und seinen Mindestwert festlegen. `ProgressIndicatorModel` verwendet ein `Adjustment`-Objekt, damit ein Threading-Zugriff auf das Datenmodell möglich ist und die Aufgabe durch ganzzahlige Werte dargestellt werden kann.

```
ProgressIndicatorModel model = new ProgressIndicatorModel(0, 100, 0);
```

In diesem Beispiel startet die Aufgabe mit dem Wert 0 und kann den Wert 100 erreichen. Diese Werte modellieren den Abschluss einer Aufgabe als Prozentsatz.

- Verbinden Sie den Controller, das Modell und die Ansicht.

```
model.setController(controller);
view.setModel(model);
view.setController(controller);
controller.setModel(model);
controller.setView(view);
```

- Erstellen Sie einen Thread, um die Aufgabe zu verarbeiten. In der Regel werden Aufgaben, die Fortschrittsanzeigen erfordern, mithilfe eines Threads ausgeführt. Aktualisieren Sie im Zuge des Aufgabenfortschritts den Wert des Modells, um den Fortschritt der Aufgabe widerzuspiegeln.

```
class ProgressThread extends Thread
{
    private boolean _paused;
    private boolean _stop;
    public void run()
    {
        // perform the task here and update the model's value as appropriate
    }
    public synchronized void setPaused(boolean paused)
    {
        // pause the indicator here
    }
    public synchronized void stopThread()
    {
        // stop the indicator here
    }
}
```

- Erstellen Sie eine Klasse, in der die `ProgressIndicatorListener`-Schnittstelle implementiert ist, die bezüglich Änderungen am Datenmodell benachrichtigt werden soll. Sie können benachrichtigt werden, wenn das Modell zurückgesetzt, wieder aufgenommen sowie abgebrochen wird oder wenn der Wert des Modells auf einem nicht programmgesteuerten Weg geändert wird.

```
private final class DemoProgressIndicatorListener implements
ProgressIndicatorListener
{
    public void cancelled()
    {
        _progressThread.setPaused(true);
    }
}
```

```

    }
    public void resumed()
    {
        _progressThread.setPaused(false);
    }
    ...
}

```

8. Verknüpfen Sie den Listener mit dem Modell.

```
model.addListener(new DemoProgressIndicatorListener());
```

9. Legen Sie die Bezeichnung für die Ansicht fest. Die Bezeichnung wird über dem ausgegebenen ProgressIndicatorField angezeigt (vorausgesetzt, es wird der Standardmanager VerticalFieldManager verwendet).

```
view.setLabel("Percent completion");
```

10. Erstellen Sie das Feld, das den Fortschritt ausgibt. Sie können Stile bereitstellen, die im ProgressBarField definiert werden, um festzulegen, ob und wie Text auf der Leiste angezeigt werden soll. Standardmäßig wird der Wert des Modells in der Mitte des Bereichs angezeigt, der von der Leiste belegt wird.

```
view.createProgressBar(Field.FIELD_HCENTER);
```

In diesem Beispiel verwendet die Fortschrittsanzeige den Standardtextstil und sie wird zentriert im Manager der Ansicht angezeigt.

11. Fügen Sie Ihrem Bildschirm die Ansicht hinzu.

```
add(view);
```

12. Steuern Sie die Fortschrittsanzeige, indem Sie bei Bedarf das Modell der Ansicht steuern.

```

MenuItem _pauseIndicator = new MenuItem("Pause indicator", 66010, 0)
{
    public void run()
    {
        view.getModel().cancel();
    }
};
MenuItem _resumeIndicator = new MenuItem("Resume indicator", 66020, 0)
{
    public void run()
    {
        view.getModel().resume();
    }
};

```

Dieses Beispiel ruft die cancel()-Methode des Modells auf, um die Fortschrittsanzeige anzuhalten. Das Beispiel ruft die resume()-Methode des Modells auf, um die Fortschrittsanzeige wieder aufzunehmen.

Codebeispiel

```

public class ProgressIndicatorScreen extends MainScreen
{
    ProgressIndicatorView view = new ProgressIndicatorView(0);
}

```

```

ProgressIndicatorModel model = new ProgressIndicatorModel(0, 100, 0);
ProgressIndicatorController controller = new ProgressIndicatorController();
ProgressThread _progressThread;
public ProgressIndicatorScreen()
{
    setTitle("Progress Indicator Screen");
    model.setController(controller);
    model.addListener(new DemoProgressIndicatorListener());
    view.setModel(model);
    view.setController(controller);
    controller.setModel(model);
    controller.setView(view);
    view.setLabel("Percent completion");
    view.createProgressBar(Field.FIELD_HCENTER);
    add(view);
    MenuItem _startIndicator = new MenuItem("Start indicator", 66000, 0)
    {
        public void run()
        {
            if(_progressThread != null)
            {
                _progressThread.stopThread();
            }
            _progressThread = new ProgressThread();
            _progressThread.start();
        }
    };
    MenuItem _pauseIndicator = new MenuItem("Pause indicator", 66010, 0)
    {
        public void run()
        {
            view.getModel().cancel();
        }
    };
    MenuItem _resumeIndicator = new MenuItem("Resume indicator", 66020, 0)
    {
        public void run()
        {
            view.getModel().resume();
        }
    };
    addMenuItem(_startIndicator);
    addMenuItem(_pauseIndicator);
    addMenuItem(_resumeIndicator);
}
class ProgressThread extends Thread
{
    private boolean _paused;
    private boolean _stop;
    public void run()
    {
        // Run dummy operations to simulate the processing of
        // a collection of data.
        for(int i = 0; i <= 100; ++i)
        {
            synchronized(this)

```

```

        {
            if(_stop)
            {
                break;
            }
            if(_paused)
            {
                try
                {
                    wait();
                }
                catch(InterruptedException ie)
                {
                }
            }
        }
        ProgressIndicatorScreen.this.model.setValue(i);
        try
        {
            // Simulate work
            sleep(250);
        }
        catch(InterruptedException ie)
        {
        }
    }
}
public synchronized void setPaused(boolean paused)
{
    _paused = paused;
    this.notify();
}
public synchronized void stopThread()
{
    _stop = true;
    if(_paused)
    {
        // If the thread is in a paused state, wake it up
        this.notify();
    }
}
}
private final class DemoProgressIndicatorListener implements
ProgressIndicatorListener
{
    public void cancelled()
    {
        _progressThread.setPaused(true);
    }
    public void resumed()
    {
        _progressThread.setPaused(false);
    }
    public void reset()
    {
        // Not implemented
    }
}

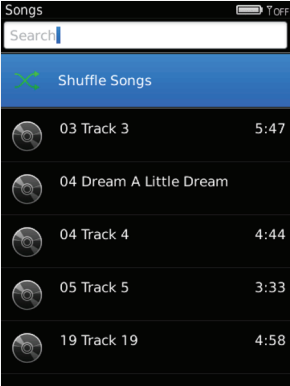
```

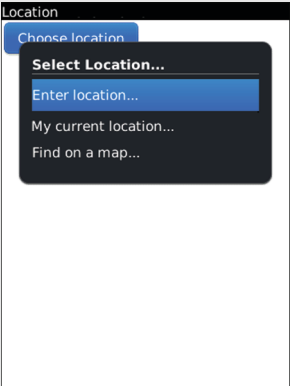
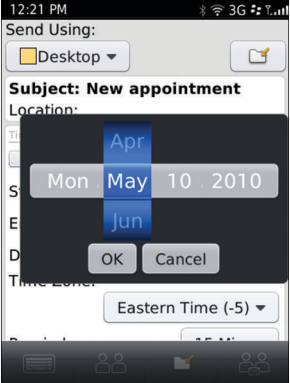
```
    }
    public void setNonProgrammaticValue(int value)
    {
        // Not implemented
    }
    public void configurationChanged(Adjustment source)
    {
        // Not implemented
    }
    public void valueChanged(Adjustment source)
    {
        // Not implemented
    }
}
}
```

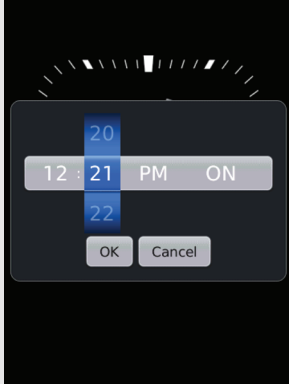
In der im BlackBerry® Java® SDK bereitgestellten Progress Indicator Demo-Beispielanwendung wird eine Fortschrittsanzeige erstellt und geändert.

Auswahltool

Sie können Auswahlen verwenden, um den Benutzern die einfache Auswahl von Elementen aus einer Liste zu ermöglichen.

Auswahltyp	Beschreibung
Ablegen	Mithilfe dieser Auswahl können Benutzer ihr BlackBerry®-Gerät nach Dateien durchsuchen. 
Ort	Mithilfe dieser Auswahl können Benutzer einen von Ihnen definierten Ort (bzw. eine Position) aus einer Liste auswählen. Zum Beispiel können Sie den Benutzern ermöglichen, ihre GPS-Position oder einen vorher selektierten Ort auszuwählen.

Auswahltyp	Beschreibung
	
Datum	Mithilfe dieser Auswahl können Benutzer einen bestimmten Tag, Monat oder ein bestimmtes Jahr auswählen. Zum Beispiel können Sie den Benutzern ermöglichen, einen Monat und ein Jahr auszuwählen, um anzugeben, wann ihre Kreditkarte abläuft. 
Uhrzeit	Mithilfe dieser Auswahl können Benutzer eine bestimmte Stunde, Minute oder Sekunde auswählen.

Auswahltyp	Beschreibung
	

Bewährtes Verfahren: Implementieren von Auswahlen

Verwenden Sie die Klassen `FilePicker`, `LocationPicker` und `DateTimePicker`, um Auswahlen zu erstellen.

Richtlinien für Dateiauswahlen

- Legen Sie eine bestimmte Ansicht fest, um die Dateitypen anzuzeigen, die mit dem Ziel des Benutzers übereinstimmen. Wenn ein Benutzer beispielsweise Bilder durchsucht, zeigen Sie in der Dateiauswahl Bilder an.
- Wenn möglich, erlauben Sie es den Benutzern, das Durchsuchen von einem entsprechenden Standardordner aus zu starten.

Erstellen einer Datumsauswahl

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.picker.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `DatePickScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```
public class DatePick extends UiApplication
{
    public static void main(String[] args)
    {
        DatePick theApp = new DatePick();
        theApp.enterEventDispatcher();
    }
}
```

```

    public DatePick()
    {
        pushScreen(new DatePickScreen());
    }
}

```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Konstruktor `setTitle()` auf, um einen Titel auf dem Bildschirm anzuzeigen. Rufen Sie `add()` auf, um ein Rich-Text-Feld auf dem Bildschirm anzuzeigen.

```

class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
    }
}

```

4. Fügen Sie der Ereigniswarteschlange der Anwendung durch Aufrufen von `invokeLater()` einen Codeabschnitt hinzu. Erstellen Sie ein `Runnable`-Objekt, und übergeben Sie es als Parameter an `invokeLater()`. Überschreiben Sie `run()` in der Definition von `Runnable`.

```

class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
            }
        });
    }
}

```

5. Rufen Sie in `run()` `DateTimePicker.getInstance()` auf, um ein `DateTimePicker`-Objekt zurückzugeben. Rufen Sie `doModal()` auf, um die Datumsauswahl anzuzeigen. Rufen Sie `getDateTime()` auf, um ein `Calendar`-Objekt zurückzugeben, das das Datum und die Uhrzeit darstellt, die der Benutzer auswählt. Verwenden Sie `getTime()`, um das Datum und die Uhrzeit als `Date`-Objekt zurückzugeben. Verwenden Sie `Dialog.alert()`, um das ausgewählte Datum und die Uhrzeit anzuzeigen.

```

class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                DateTimePicker datePicker = DateTimePicker.getInstance();
            }
        });
    }
}

```

```

        datePicker.doModal();
        Calendar cal = datePicker.getDateTime();
        Date date = cal.getTime();
        Dialog.alert("You selected " + date.toString());
    }
});
}
}

```

Codebeispiel: Erstellen einer Datumsauswahl

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
import java.util.*;
public class DatePick extends UiApplication
{
    public static void main(String[] args)
    {
        DatePick theApp = new DatePick();
        theApp.enterEventDispatcher();
    }
    public DatePick()
    {
        pushScreen(new DatePickScreen());
    }
}
class DatePickScreen extends MainScreen
{
    public DatePickScreen()
    {
        setTitle("Date Picker Sample");
        add(new RichTextField("Trying Date Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                DateTimePicker datePicker = DateTimePicker.getInstance();
                datePicker.doModal();
                Calendar cal = datePicker.getDateTime();
                Date date = cal.getTime();
                Dialog.alert("You selected " + date.toString());
            }
        });
    }
}
}

```

Erstellen einer Dateiauswahl

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.util.*;
```

- Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `FilePickScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```
public class FilePick extends UiApplication
{
    public static void main(String[] args)
    {
        FilePick theApp = new FilePick();
        theApp.enterEventDispatcher();
    }
    public FilePick()
    {
        pushScreen(new FilePickScreen());
    }
}
```

- Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um einen Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um dem Bildschirm ein Bezeichnungsfeld hinzuzufügen.

```
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
    }
}
```

- Rufen Sie im Bildschirmkonstruktor `invokeLater()` auf, um der Ereigniswarteschlange der Anwendung einen Codeabschnitt hinzuzufügen. Erstellen Sie ein `Runnable`-Objekt, und übergeben Sie es als Parameter an `invokeLater()`.

```
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
            }
        });
    }
}
```

```

    });
}

```

5. Überschreiben Sie `run()` in der Definition von `Runnable`. Rufen Sie in `run()` `FilePicker.getInstance()` auf, um ein `FilePicker`-Objekt zurückzugeben. Erstellen Sie ein `FilePickListener`-Objekt, das in Schritt 6 beschrieben wird. Rufen Sie `setListener()` auf, um den Listener für die Dateiauswahl zu registrieren. Rufen Sie `show()` auf, um die Dateiauswahl auf dem Bildschirm anzuzeigen.

```

class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                FilePicker fp = FilePicker.getInstance();
                FilePickListener fileListener = new FilePickListener();
                fp.setListener(fileListener);
                fp.show();
            }
        });
    }
}

```

6. Rufen Sie `Dialog.alert()` auf, um ein Dialogfeld mit einer Meldung zu erstellen, die anzeigt, welche Datei ausgewählt wurde. Rufen Sie `Dialog.alert()` in einer Klasse auf, die die `FilePicker.Listener`-Schnittstelle durch Überschreiben von `selectionDone()` implementiert. Die Anwendung ruft `selectionDone()` auf, wenn der Benutzer mithilfe der Dateiauswahl eine Datei auswählt.

```

class FilePickListener implements FilePicker.Listener
{
    public void selectionDone(String str)
    {
        Dialog.alert("You selected " + str);
    }
}

```

Codebeispiel: Erstellen eines Dateipickers

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.picker.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.io.*;
public class FilePick extends UiApplication
{
    public static void main(String[] args)
    {

```

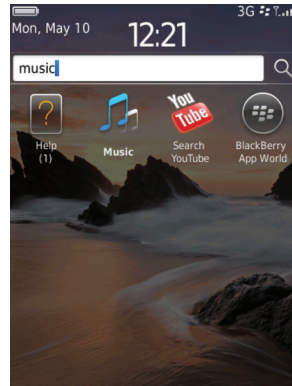
```

        FilePick theApp = new FilePick();
        theApp.enterEventDispatcher();
    }
    public FilePick()
    {
        pushScreen(new FilePickScreen());
    }
}
class FilePickScreen extends MainScreen
{
    public FilePickScreen()
    {
        setTitle("File Picker Sample");
        add(new LabelField("Trying File Picker"));
        UiApplication.getUiApplication().invokeLater(new Runnable()
        {
            public void run()
            {
                FilePicker fp = FilePicker.getInstance();
                FilePickListener fileListener = new FilePickListener();
                fp.setListener(fileListener);
                fp.show();
            }
        });
    }
}
class FilePickListener implements FilePicker.Listener
{
    public void selectionDone(String str)
    {
        Dialog.alert("You selected " + str);
    }
}

```

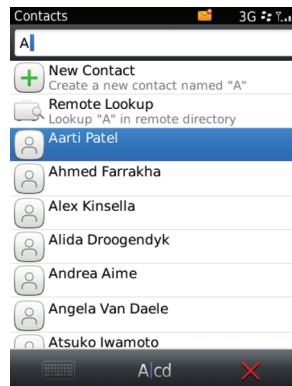
Suche

Benutzer können das Suchfeld auf der Startseite verwenden, um nach Elementen in einer beliebigen Anwendung auf dem Gerät zu suchen. Dazu gehören auch Drittanbieteranwendungen. Die Suche kann auch Inhalt enthalten, der nicht auf dem Gerät gespeichert ist, wie z. B. die Datenbank eines Unternehmens oder eine Website.



Benutzer können auch ein Suchfeld in einer Anwendung verwenden, um nach Elementen in dieser Anwendung zu suchen. Zum Beispiel könnten Benutzer eine E-Mail-Nachricht in einer Nachrichtenliste, ein Lied in der Medienanwendung oder einen Kontakt in der Kontaktliste suchen. In einigen Fällen möchten Sie eventuell Suchergebnisse aus anderen Anwendungen anzeigen.

In einigen Anwendungen wird das Suchfeld auf dem Bildschirm angezeigt. In anderen Fällen ist die Suche im vollständigen Menü, Popup-Menü oder in der Werkzeugleiste verfügbar. Zeigen Sie die Suchergebnisse an, noch während die Benutzer Text in ein Suchfeld eingeben. Wenn eine große Anzahl an Suchergebnissen zurückgegeben wird, ermöglichen Sie es den Benutzern, ihre Suche auf ein Feld oder eine Gruppe von Feldern einzugrenzen. Wenn die Benutzer zum Beispiel die Nachrichtenliste durchsuchen, können sie die Dropdown-Liste rechts vom Suchfeld verwenden, um ihre Suche auf das Feld "An" oder "Betreff" einzugrenzen. Weitere Informationen zum Hinzufügen von Suchfeldern zu Ihrer Anwendung finden Sie im *BlackBerry Java Application Integration Development Guide*.



Benutzer können in einem Suchfeld folgende Aktionen ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Öffnen eines markierten Elements in den Suchergebnissen.	• Drücken Sie die Eingabetaste. • Klicken mit dem Trackpad.	• Antippen des Bildschirms. • Drücken Sie die Eingabetaste. • Klicken mit dem Trackpad.
Anzeigen eines Popup-Menüs mit Aktionen für ein Suchergebnis (zum Beispiel das Anrufen eines Kontakts)	Klicken und halten Sie mit dem Trackpad.	• Berühren und halten Sie ein Suchergebnis auf dem Bildschirm. • Klicken und halten Sie mit dem Trackpad.

Sie können Inhalte in Ihrer Anwendung registrieren, damit diese in den Suchergebnissen aufgenommen werden können. Sie können außerdem Ihre Anwendung registrieren, um den Benutzern die Möglichkeit zu geben, ihre Suche zu erweitern. Wenn die Benutzer zum Beispiel nach einem Lied in der Medienanwendung suchen, und das Lied dort nicht finden, können Sie den Benutzern das Durchsuchen Ihrer Anwendung erlauben und so eine alternative Quelle für die Suchergebnisse bereitstellen. Weitere Informationen zum Registrieren von Inhalten oder einer Anwendung finden Sie im *BlackBerry Java Application Integration Development Guide*.

Bewährtes Verfahren: Implementieren von Suchfunktionen

- Verwenden Sie das `net.rim.device.api.unifiedsearch`-Paket, um Suchfunktionen zu implementieren.
- Selektieren Sie gezielt die Inhalte, die Sie zur Anzeige in den Suchergebnissen registrieren. Registrieren Sie nur die Inhalte, die den Benutzern bedeutungsvolle Suchergebnisse bereitstellen.
- Versuchen Sie, die relevantesten Elemente am Anfang der Suchergebnisliste anzuzeigen. Wenn Benutzer beispielsweise ein Restaurant suchen, das an mehreren verschiedenen Standorten zu finden ist, zeigen Sie am Anfang der Suchergebnisliste das Restaurant an, das sich am nächsten zum Benutzerstandort befindet.
- Markieren Sie in den Suchergebnissen den Text in Fettschrift, der mit dem vom Benutzer eingegebenen Text übereinstimmt. Mit diesem Ansatz können die Benutzer erkennen, warum jedes Element in der Suchergebnisliste angezeigt wird.
- Wenn Benutzer auf einem Bildschirm nach einem Wort suchen müssen (zum Beispiel in einer Nachricht oder auf einer Webseite), verwenden Sie im Menü den Begriff "Suchen".

Implementieren von Suchfeldern

- Setzen Sie den Standardfokus auf das Suchfeld. Wenn die Suchergebnisse angezeigt werden, setzen Sie den Standardfokus auf das erste Element in der Liste der Suchergebnisse.
- Verwenden Sie den Begriff "Suchen" als Hinweistext im Suchfeld.
- Machen Sie Suchfelder nicht von Groß- und Kleinschreibung abhängig.
- Platzieren Sie das Suchfeld unter die Titelleiste eines Anwendungsbildschirms.
- Weisen Sie einem Bildschirm, der ein Suchfeld enthält, keine Tastenkombinationen zu. Wenn Sie Tastenkombinationen verwenden möchten, stellen Sie alternative Suchfunktionen bereit. Erlauben Sie zum Beispiel Benutzern, nach Nachrichten in einer Nachrichtenliste zu suchen, indem sie das Menü verwenden.

Erstellen eines Suchfelds

Sie können eine Anwendung erstellen, die die `KeywordFilterField`-Klasse verwendet, die im `net.rim.device.api.ui.component`-Paket enthalten ist, um ein Benutzeroberflächenfeld bereitzustellen, das aus einem einzelnen Texteingabefeld und einer Liste auswählbarer Elemente besteht. Während die Benutzer Text in ein Suchfeld eingeben, filtert die Anwendung die Elemente in der Liste, die mit dem Suchtext beginnen. Weitere Informationen zur Verwendung der `KeywordFilterField`-Klasse finden Sie in der Beispielanwendung zum Stichwortfilterfeld, die in der BlackBerry® Java® Development Environment Version 4.3.1 oder höher enthalten ist.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.collection.util.SortedReadableList;
import net.rim.device.api.io.LineReader;
import net.rim.device.api.ui.component.KeywordFilterField;
import net.rim.device.api.ui.component.KeywordProvider;
import java.io.InputStream;
import java.lang.String;
import java.util.Vector;
```

2. Erstellen Sie Variablen. Im folgenden Codebeispiel erweitert `CountryList` die `SortedReadableList`-Klasse und implementiert die `KeywordProvider`-Schnittstelle.

```
private KeywordFilterField _keywordField;
private CountryList _countryList;
private Vector _countries;
```

3. Um eine Liste auswählbarer Textelemente zu erstellen, füllen Sie einen Vektor mit Daten aus einer Textdatei.

```
_countries = getDataFromFile();
```

4. Erstellen Sie eine Instanz einer Klasse, die die `SortedReadableList`-Klasse erweitert.

```
_countryList = new
    CountryList(StringComparator.getInstance(true), _countries);
```

5. Um die Elemente der Liste anzugeben, erstellen Sie eine neue Instanz eines `KeywordFilterField`-Objekts.

```
_keywordField = new KeywordFilterField();
```

6. Rufen Sie `KeywordFilterField.setList()` auf.

```
_keywordField.setList(_countryList, _countryList);
```

7. Legen Sie eine Bezeichnung für das Eingabefeld von `KeywordFilterField` fest.

```
_keywordField.setLabel("Search: ");
```

8. Erstellen Sie den Hauptbildschirm der Anwendung, und fügen Sie dem Hauptbildschirm ein `KeywordFilterField`-Objekt hinzu.

```
KeywordFilterDemoScreen screen = new
    KeywordFilterDemoScreen(this, _keywordField);
screen.add(_keywordField.getKeywordField());
screen.add(_keywordField);
pushScreen(screen);
```

9. Um eine Methode zu erstellen, die einen Vektor von Country-Objekten füllt und zurückgibt, die Daten aus der Textdatei enthalten, geben Sie in der Methodensignatur als Rückgabotyp Vector an.

```
public Vector getDataFromFile()
```

10. Erstellen und speichern Sie einen Verweis auf ein neues Vector-Objekt.

```
Vector countries = new Vector();
```

11. Erstellen Sie einen Eingabestrom zur Textdatei.

```
InputStream stream =  
    getClass().getResourceAsStream("/Data/CountryData.txt");
```

12. Lesen Sie durch CRLF getrennte Zeilen aus dem Eingabestrom.

```
LineReader lineReader = new LineReader(stream);
```

13. Lesen Sie Zeile für Zeile Daten aus dem Eingabestrom, bis Sie die Markierung für das Dateiende erreichen. Jede Zeile wird analysiert, um Daten zu extrahieren, die für das Erstellen von Country-Objekten verwendet werden.

```
for(;;){  
    //Obtain a line of text from the text file  
    String line = new String(lineReader.readLine());  
    //If we are not at the end of the file, parse the line of text  
    if(!line.equals("EOF")) {  
        int space1 = line.indexOf(" ");  
        String country = line.substring(0,space1);  
        int space2 = line.indexOf(" ",space1+1);  
        String population = line.substring(space1+1,space2);  
        String capital = line.substring(space2+1,line.length());  
        // Create a new Country object  
        countries.addElement(new Country(country,population,capital));  
    }  
    else {  
        break;  
    }  
} // end the for loop  
return countries;
```

14. Um der Liste auswählbarer Textelemente ein Stichwort hinzuzufügen, rufen Sie SortedReadableList.doAdd(element) auf.

```
SortedReadableList.doAdd(((Country)countries.elementAt(i))  
    .getCountryName());
```

15. Um die Liste auswählbarer Textelemente zu aktualisieren, rufen Sie KeywordFilterField.updateList() auf.

```
_keywordField.updateList();
```

16. Um das Stichwort zu erhalten, das ein BlackBerry-Gerätebenutzer in das KeywordFilterField-Feld eingegeben hat, rufen Sie KeywordFilterField.getKeyword() auf.

```
String userTypedWord = _keywordField.getKeyword();
```

Autocomplete-Textfeld

Sie können ein autocomplete-Textfeld verwenden, um vorherzusagen, was ein BlackBerry®-Gerätebenutzer eingeben möchte, und ein Wort bzw. einen Satz anzuzeigen, bevor der Benutzer dessen Eingabe abgeschlossen hat.

Wenn Sie ein `AutoCompleteField`-Objekt erstellen, müssen Sie ein `BasicFilteredList`-Objekt damit verknüpfen. Durch `BasicFilteredList` bleiben die Bezüge auf Datenobjekte erhalten, mit denen verglichen wird, um die Liste der Wörter und Sätze zu erstellen. Sie können in der Konfiguration festlegen, mit welchen Feldern in den Datenobjekten verglichen wird, und welche Felder angezeigt werden, wenn ein Gegenstück gefunden wird. So können Sie beispielsweise den Text, den der Benutzer eingibt, mit dem Wert des `DATA_FIELD_CONTACTS_BIRTHDAY`-Felds in der `DATA_SOURCE_CONTACTS`-Datenquelle vergleichen, und den Wert des entsprechenden `DATA_FIELD_CONTACTS_NAME_FULL`-Felds zurückgeben.

Es gibt vier Datentypen, die Sie an eine `BasicFilteredList` binden können, um diese mit einem `AutoCompleteField` zu verwenden.

Sie können den Satz der Zeichenfolgen angeben, mit dem auf eine der folgenden Arten verglichen wird:

- Ein Array wörtlicher Zeichenfolgen
- Ein Array von Objekten, das `toString()` unterstützt
- Datenquellen auf einem BlackBerry-Gerät, wie z. B. Kontakte, Notizen, Aufgaben und verschiedene Arten von Mediendateien
- Ein Array von Objekten und einem Array von Zeichenfolgen mit entsprechenden Indizes

Standardmäßig zeigt das autocomplete-Textfeld den Satz der Zeichenfolgen an, der vom Vergleichsprozess in einer Dropdown-Liste zurückgegeben wird. Sie können das Erscheinungsbild dieser Liste durch Angeben von Stilmarkierungen bei der Erstellung des autocomplete-Textfeldes konfigurieren. Sie können ändern, wie die Liste angezeigt wird, und wie Benutzer mit der Liste interagieren können.

Erstellen eines autocomplete-Textfelds aus einem Datensatz

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `HomeScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
}
```

```

    }
    AutoCompleteFieldApp()
    {
        pushScreen(new HomeScreen());
    }
}

```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der mainScreen-Klasse.

```

class HomeScreen extends mainScreen
{
    public HomeScreen()
    {
    }
}

```

4. Erstellen Sie im Konstruktor ein BasicFilteredList-Objekt. Erstellen Sie ein String-Array, und speichern Sie die Zeichenfolgen, die sich im Array entsprechen sollen. In diesem Beispiel sind die Zeichenfolgen Wochentage. Rufen Sie addDataSet() auf, um die Daten im Array an BasicFilteredList zu binden.

```

BasicFilteredList filterList = new BasicFilteredList();
String[] days =
{"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
filterList.addDataSet(1, days, "days", BasicFilteredList.COMPARISON_IGNORE_CASE);

```

5. Erstellen Sie im Konstruktor ein AutoCompleteField-Objekt. Übergeben Sie eine Instanz von BasicFilteredList an den AutoCompleteField-Konstruktor, um BasicFilteredList an das autocomplete-Textfeld zu binden. Rufen Sie add() auf, um das Feld dem Bildschirm hinzuzufügen.

```

AutoCompleteField autoCompleteField = new
    AutoCompleteField(filterList);
add(autoCompleteField);

```

Codebeispiel: Erstellen eines autocomplete-Felds aus einem Datensatz

```

import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    AutoCompleteFieldApp()
    {
        HomeScreen scr = new HomeScreen();
        this.pushScreen(scr);
    }
}
class HomeScreen extends mainScreen
{

```

```

public HomeScreen()
{
    BasicFilteredList filterList = new BasicFilteredList();
    String[] days = {"Monday","Tuesday","Wednesday",
                    "Thursday","Friday","Saturday","Sunday"};
    filterList.addDataSet(1,days,"days",BasicFilteredList
.COMPARISON_IGNORE_CASE);
    autoCompleteField autoCompleteField = new autoCompleteField(filterList);
    add(autoCompleteField);
}
}

```

Erstellen eines autocomplete-Textfelds aus einer Datenquelle

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;

```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `HomeScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```

public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    public AutoCompleteFieldApp()
    {
        pushScreen(new HomeScreen());
    }
}

```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse.

```

class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
    }
}

```

4. Erstellen Sie im Bildschirmkonstruktor ein `BasicFilteredList`-Objekt. Rufen Sie `addDataSource()` auf, um eine Datenquelle an `BasicFilteredList` zu binden. In diesem Beispiel handelt es sich bei den Daten um Kontaktinformationen und die Datenquelle ist die Kontaktliste. Das erste Argument, das Sie an `addDataSource()` übergeben, ist eine eindeutige ID. Das zweite Argument bindet das `BasicFilteredList` an eine Datenquelle. Das dritte Argument gibt den Satz von Datenquellenfeldern an, mit denen verglichen wird. Das

vierte Argument gibt den Satz von Datenquellenfeldern an, die bereitgestellt werden sollen, wenn eine Übereinstimmung gefunden wird. In diesem Beispiel sind die Felder, mit denen verglichen wird, identisch mit den Feldern, die bereitgestellt werden sollen, wenn eine Übereinstimmung gefunden wird. Das fünfte Argument gibt das primäre Anzeigefeld an. Das sechste Argument gibt das sekundäre Anzeigefeld an und ist auf -1 gesetzt, wenn Sie es nicht verwenden möchten. Das letzte Argument gibt einen Namen für `BasicFilteredList` an. Sein Wert wird automatisch generiert, wenn Sie null angeben.

```
BasicFilteredList filterList = new BasicFilteredList();
    filterList.addDataSource(
        1,
        BasicFilteredList.DATA_SOURCE_CONTACTS,
        BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
        BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
        BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
        BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
        BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
        BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
        BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
        -1,
        null);
```

5. Erstellen Sie im Bildschirmkonstruktor ein `AutoCompleteField`-Objekt. Übergeben Sie das `BasicFilteredList`-Objekt, das Sie in Schritt 4 erstellt haben, an den `AutoCompleteField`-Konstruktor, um `BasicFilteredList` an das `autocomplete`-Textfeld zu binden. Rufen Sie `add()` auf, um das Feld dem Bildschirm hinzuzufügen.

```
AutoCompleteField autoCompleteField = new
    AutoCompleteField(filterList);
add(autoCompleteField);
```

Codebeispiel: Erstellen eines autocomplete-Felds aus einer Datenquelle

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.component.AutoCompleteField;
import net.rim.device.api.collection.util.*;
public class AutoCompleteFieldApp extends UiApplication
{
    public static void main(String[] args)
    {
        AutoCompleteFieldApp app = new AutoCompleteFieldApp();
        app.enterEventDispatcher();
    }
    AutoCompleteFieldApp()
    {
        HomeScreen scr = new HomeScreen();
        this.pushScreen(scr);
    }
}
class HomeScreen extends MainScreen
{
    public HomeScreen()
    {
        BasicFilteredList filterList = new BasicFilteredList();
```

```
        filterList.addDataSource(1,
            BasicFilteredList.DATA_SOURCE_CONTACTS,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL |
BasicFilteredList.DATA_FIELD_CONTACTS_COMPANY |
BasicFilteredList.DATA_FIELD_CONTACTS_EMAIL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
            BasicFilteredList.DATA_FIELD_CONTACTS_NAME_FULL,
            null);
        autoCompleteField autoCompleteField = new autoCompleteField(filterList);
        add(autoCompleteField);
    }
}
```

Verwenden von Datenquellen und Feldern mit einem autocomplete-Textfeld

Mithilfe der Klassen autoCompleteField und BasicFilteredList können Sie den Text, den ein Benutzer in einem autocomplete-Textfeld eingibt, mit den Werten von Feldern in einer angegebenen Datenquelle vergleichen. Sie geben die zu verwendenden Felder und ihre Datenquellen mithilfe eines BasicFilteredList-Objekts an, das Sie als Argument an den Konstruktor der autoCompleteField-Klasse übergeben.

Datenquelle	Felder
DATA_SOURCE_APPOINTMENTS	- DATA_FIELD_APPOINTMENTS_ALL - DATA_FIELD_APPOINTMENTS_ATTENDEES - DATA_FIELD_APPOINTMENTS_ORGANIZER - DATA_FIELD_APPOINTMENTS_SUBJECT
DATA_SOURCE_CONTACTS	- DATA_FIELD_CONTACTS_ADDRESS_ALL - DATA_FIELD_CONTACTS_ADDRESS_HOME - DATA_FIELD_CONTACTS_ADDRESS_WORK - DATA_FIELD_CONTACTS_ANNIVERSARY - DATA_FIELD_CONTACTS_BIRTHDAY - DATA_FIELD_CONTACTS_CATEGORIES - DATA_FIELD_CONTACTS_COMPANY - DATA_FIELD_CONTACTS_EMAIL - DATA_FIELD_CONTACTS_FAX - DATA_FIELD_CONTACTS_JOB_TITLE - DATA_FIELD_CONTACTS_NAME_FULL - DATA_FIELD_CONTACTS_NAME_FIRST - DATA_FIELD_CONTACTS_NAME_LAST - DATA_FIELD_CONTACTS_NOTES - DATA_FIELD_CONTACTS_PAGER - DATA_FIELD_CONTACTS_PHONE_ALL

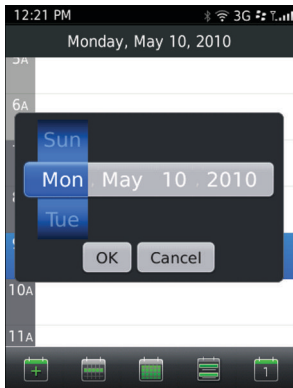
Datenquelle	Felder
	- DATA_FIELD_CONTACTS_PHONE_HOME - DATA_FIELD_CONTACTS_PHONE_HOME2 - DATA_FIELD_CONTACTS_PHONE_MOBILE - DATA_FIELD_CONTACTS_PHONE_OTHER - DATA_FIELD_CONTACTS_PHONE_WORK - DATA_FIELD_CONTACTS_PHONE_WORK2 - DATA_FIELD_CONTACTS_PIN
DATA_SOURCE_MEMOS	DATA_FIELD_MEMOS_TITLE
DATA_SOURCE_MESSAGES	- DATA_FIELD_MESSAGES_ALL - DATA_FIELD_MESSAGES_RECIPIENT - DATA_FIELD_MESSAGES_SENDER - DATA_FIELD_MESSAGES_SUBJECT
DATA_SOURCE_MUSIC	- DATA_FIELD_MUSIC_ALL - DATA_FIELD_MUSIC_ALBUM - DATA_FIELD_MUSIC_ARTIST - DATA_FIELD_MUSIC_GENRE - DATA_FIELD_MUSIC_PLAYLIST - DATA_FIELD_MUSIC_SONG
DATA_SOURCE_PICTURES	DATA_FIELD_PICTURES_TITLE
DATA_SOURCE_RINGTONES	DATA_FIELD_RINGTONES_TITLE
DATA_SOURCE_TASKS	DATA_FIELD_TASKS_TITLE
DATA_SOURCE_VIDEOS	DATA_FIELD_VIDEOS_TITLE
DATA_SOURCE_VOICENOTES	DATA_FIELD_VOICENOTES_TITLE

Drehfelder

Verwenden Sie ein Drehungsfeld, um Benutzern die problemlose Auswahl eines Elements aus einer sortierten Liste zu ermöglichen. Verwenden Sie beispielsweise Drehungsfelder, um es Benutzern zu ermöglichen, eine Zahl zu finden oder den Wochentag zu ändern.

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Ein Element in der Liste finden.	Fahren Sie mit einem Finger senkrecht über das Trackpad.	- Ziehen Sie einen Finger senkrecht über den Bildschirm. - Führen Sie den Finger auf dem Bildschirm nach oben oder nach unten.

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
		Fahren Sie mit einem Finger senkrecht über das Trackpad. Fahren Sie mit dem Finger auf dem Bildschirm oder Trackpad nach oben oder nach unten.
Element aus der Liste auswählen.	Klicken mit dem Trackpad.	- Nehmen Sie einen Finger weg vom Bildschirm. - Klicken mit dem Trackpad.
Zu einem anderen Drehungsfeld wechseln.	Fahren Sie mit einem Finger senkrecht über das Trackpad.	- Ziehen Sie einen Finger senkrecht über den Bildschirm. - Fahren Sie mit einem Finger senkrecht über das Trackpad.



Erstellen eines Drehfelds

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.component.Dialog;
import net.rim.device.api.ui.component.TextSpinBoxField;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.container.SpinnerFieldManager;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungsconstructor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die `HomeScreen`-Klasse stellt den in Schritt 3 beschriebenen benutzerdefinierten Bildschirm dar.

```
public class SpinBoxApp extends UiApplication
{
    public static void main(String[] args)
    {
        SpinBoxApp app = new SpinBoxApp();
        app.enterEventDispatcher();
    }
    public SpinBoxApp()
    {
        pushScreen(new HomeScreen());
    }
}
```

- Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der MainScreen-Klasse. Deklarieren Sie eine Variable für jedes Feld im Drehfeld sowie eine Variable für den Drehfeld-Manager.

```
class HomeScreen extends MainScreen
{
    TextSpinBoxField spinBoxDays;
    TextSpinBoxField spinBoxMonths;
    SpinBoxFieldManager spinBoxMgr;
    public HomeScreen()
    {
    }
}
```

- Erstellen Sie im Bildschirmkonstruktor für jedes der Drehfelder ein Array von String-Objekten.

```
final String[] DAYS =
    {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
final String[] MONTHS =
    {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
```

- Erstellen Sie im Bildschirmkonstruktor neue Instanzen des Drehfeld-Managers sowie der zwei Drehfelder. Übergeben Sie das entsprechende Array von Zeichenfolgen als Argumente an jeden der Drehfeldkonstruktoren.

```
spinBoxMgr = new SpinBoxFieldManager();
spinBoxDays = new TextSpinBoxField(DAYS);
spinBoxMonths = new TextSpinBoxField(MONTHS);
```

- Fügen Sie im Bildschirmkonstruktor dem Drehfeld-Manager die Drehfelder hinzu. Rufen Sie add() auf, um den Manager sowie die Felder, die er enthält, dem Bildschirm hinzuzufügen.

```
spinBoxMgr.add(spinBoxDays);
spinBoxMgr.add(spinBoxMonths);
add(spinBoxMgr);
```

Codebeispiel: Erstellen eines Drehfelds

```
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.container.MainScreen;
import net.rim.device.api.ui.container.SpinBoxFieldManager;
import net.rim.device.api.ui.component.Dialog;
import net.rim.device.api.ui.component.TextSpinBoxField;
```

```

public class SpinBoxApp extends UiApplication
{
    public static void main(String[] args)
    {
        SpinBoxApp app = new SpinBoxApp();
        app.enterEventDispatcher();
    }
    public SpinBoxApp()
    {
        HomeScreen homeScreen = new HomeScreen();
        pushScreen(homeScreen);
    }
}
class HomeScreen extends MainScreen
{
    TextSpinBoxField spinBoxDays;
    TextSpinBoxField spinBoxMonths;
    SpinBoxFieldManager spinBoxMgr;
    public HomeScreen()
    {
        final String[] DAYS =
        {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
        final String[] MONTHS =
        {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
        spinBoxMgr = new SpinBoxFieldManager();
        spinBoxDays = new TextSpinBoxField(DAYS);
        spinBoxMonths = new TextSpinBoxField(MONTHS);
        spinBoxMgr.add(spinBoxDays);
        spinBoxMgr.add(spinBoxMonths);
        add(spinBoxMgr);
    }
    public void close()
    {
        Dialog.alert("You selected " +
        (String)spinBoxDays.get(spinBoxDays.getSelectedIndex()) + " and " +
        (String)spinBoxMonths.get(spinBoxMonths.getSelectedIndex()));
        super.close();
    }
}

```

Bewährtes Verfahren: Implementieren von Drehfeldern

- Verwenden Sie Drehfelder bei einer Liste sequenzieller Elemente.
- Verwenden Sie Dropdown-Listen bei nicht sequenziellen Elementen oder Elementen mit unregelmäßigen Abständen. Bei einer kurzen Liste nicht sequenzieller Elemente können Sie ein Drehfeld verwenden, um eine interaktivere Funktionalität für Benutzer bereitzustellen.
- Vermeiden Sie die Verwendung eines Drehfelds, wenn mehrere andere Komponenten auf dem Bildschirm angezeigt werden.
- Verwenden Sie die Klassen `SpinBoxField` und `SpinBoxFieldManager`, um Drehfelder zu erstellen.
- Fügen Sie Drehfelder möglichst Dialogfeldern statt Bildschirmen hinzu.
- Wenn Benutzer ein Drehfeld markieren, zeigen Sie drei bis fünf Elemente übereinander an.

- Verwenden Sie ein erkennbares Muster für die Abfolge der Elemente (zum Beispiel 5, 10, 15), damit Benutzer abschätzen können, wie weit sie scrollen müssen, um das Zielelement zu finden.
- Vermeiden Sie, Benutzer horizontal scrollen zu lassen, um mehrere Drehfelder anzuzeigen. Unterteilen Sie Drehfelder falls notwendig in mehrere Felder.
- Wenn der Text in einem Drehfeld zu lang ist, verwenden Sie Auslassungspunkte (...).

Textfelder

Benutzer können ein Textfeld zur Eingabe von Text verwenden.

Art des Textfelds	Beschreibung
E-Mail	Zum Einfügen des Zeichens "@" oder eines Punktes (.) in ein E-Mail-Adressenfeld drücken Benutzer die Leertaste.
Datum und Uhrzeit	Benutzer können das Datum oder die Zeit auf BlackBerry®-Geräten mit Trackpad ändern, indem Sie die Tastatur verwenden oder mit einem Finger senkrecht über das Trackpad fahren. Benutzer können das Datum oder die Zeit auf BlackBerry-Geräten mit Touchscreen ändern, indem Sie den Finger auf dem Bildschirm nach oben oder nach unten führen.
Nummer	Müssen Benutzer ein Nummernfeld auf einer physischen Tastatur eingeben, wechselt das BlackBerry-Gerät zum Modus Zahlensperre, sodass die Benutzer die Alt-Taste zum Eingeben von Zahlen nicht drücken müssen. Müssen Benutzer ein Nummernfeld auf einer virtuellen Tastatur eingeben, wird die Zahlentastatur angezeigt.
Kennwort	Wenn Benutzer Text in ein Kennwortfeld eingeben, werden anstelle des Textes Sternchen (*) angezeigt. In Kennwortfeldern können Benutzer keinen Text ausschneiden, kopieren oder einfügen, bzw. AutoText nicht verwenden. Bei BlackBerry-Geräten mit SureType®-Technologie ist Mehrfachtippen die standardmäßige Eingabemethode in Kennwortfeldern.
Telefonnummer	Geben Benutzer ein Feld "Telefonnummer " auf einer physischen Tastatur ein, wechselt das BlackBerry-Gerät zum Modus Zahlensperre, sodass die Benutzer die Alt-Taste zum Eingeben von Zahlen nicht drücken müssen. Sie können auch zulassen, dass Benutzer in Telefonnummernfeldern die folgenden Aktionen ausführen: • Eingeben des Pluszeichens (+) für internationale Telefonnummern. • Eingeben von Formatierungszeichen, z. B. das Minuszeichen (-), den Punkt (.), Klammern (()) und Leerzeichen. • Eingeben eines Nummernzeichens (#) oder Sternchens (*). • Anzeigen einer Pause durch Eingeben eines Kommas (,) oder Anzeigen einer Wartezeit durch Eingeben eines Ausrufezeichens (!). • Anzeigen einer Durchwahl durch Drücken der Alt-Taste und Drücken von E, X oder T.

Art des Textfelds	Beschreibung
	Müssen Benutzer ein Feld "Telefonnummer " auf einer virtuellen Tastatur eingeben, wird die Zehntastatur angezeigt. Sie können auch zulassen, dass Benutzer in Telefonnummernfeldern die folgenden Aktionen ausführen: • Eingeben eines Nummernzeichens (#) oder Sternchens (*). • Anzeigen einer Pause oder einer Durchwahl durch Gedrückthalten der Sternchen-Taste (*). • Anzeigen einer Wartezeit oder einer Durchwahl durch Gedrückthalten der Nummernzeichen-Taste (#).
Text	In Textfeldern geben Benutzer Text ein. Benutzer können in Textfeldern Text ausschneiden, kopieren und einfügen. Wenn der Cursor das Ende einer Textzeile erreicht, erfolgt ein Zeilenumbruch in die nächste Zeile. In Textfeldern können BlackBerry-Geräte Telefonnummern, Webseiten und E-Mail-Adressen auch automatisch in Links umwandeln.
Webadresse	Benutzer können durch Drücken der Leertaste einen Punkt (.) in ein Adressfeld einfügen.

Bewährtes Verfahren: Implementieren von Textfeldern

- Verwenden Sie die `TextField`-Klasse, um Textfelder zu erstellen.
- Wählen Sie den Typ des Textfelds basierend darauf aus, welche Art von Eingabe Sie von den Benutzern erwarten. Die Verwendung eines geeigneten Textfelds ist wichtig, damit die entsprechende Eingabeanzeige auf dem Bildschirm angezeigt wird, wenn Benutzer Text eingeben. Wenn Benutzer zum Beispiel Text in ein Feld für Telefonnummern eingeben, wird in der rechten oberen Ecke des Bildschirms die Zahlensperrenanzeige angezeigt. Das von Ihnen ausgewählte Feld beeinflusst auch die Standardeingabemethode für das Feld auf BlackBerry-Geräten mit SureType®-Technologie.
- Wenn möglich, verwenden oder erweitern Sie vorhandene Felder statt benutzerdefinierte Felder zu erstellen, sodass die Felder das entsprechende Standardverhalten erben.
- Fügen Sie ggf. einen Hinweistext als Anleitung für die Benutzer hinzu. Verwenden Sie einen Hinweistext, wenn nicht genügend Platz auf dem Bildschirm vorhanden ist, um eine Bezeichnung oder eine Anweisung anzuzeigen. Der Hinweistext wird im Feld angezeigt und ausgeblendet, sobald die Benutzer Eingaben vornehmen.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze, klare beschreibende Bezeichnungen. Vermeiden Sie Bezeichnungen mit Zeilenumbrüchen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Titel.
- Schließen Sie die Bezeichnungen von Feldern mit einem Doppelpunkt (:) ab.
- Wenn Sie einen Hinweistext verwenden, formulieren Sie ihn kurz und klar. Verwenden Sie die übliche Groß-/Kleinschreibung für Sätze.

Erstellen eines Textfelds

Erstellen eines schreibgeschützten Textfelds, das eine Formatierung ermöglicht

1. Importieren Sie die `net.rim.device.api.ui.component.RichTextField`-Klasse.
2. Erstellen Sie eine Instanz von `RichTextField`.

```
RichTextField rich = new RichTextField("RichTextField");
```

Erstellen eines bearbeitbaren Textfelds, das keine Formatierung besitzt und Filter akzeptiert

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.component.BasicEditField`
 - `net.rim.device.api.ui.component.EditField`
2. Erstellen Sie eine Instanz von `BasicEditField`.

```
BasicEditField bf = new BasicEditField("BasicEditField: ", "", 10, EditField.FILTER_UPPERCASE);
```

Erstellen eines bearbeitbaren Textfelds, das Sonderzeichen ermöglicht

1. Importieren Sie die `net.rim.device.api.ui.component.EditField`-Klasse.
2. Erstellen Sie eine Instanz von `EditField`.

```
EditField edit = new EditField("EditField: ", "", 10, EditField.FILTER_DEFAULT);
```

Erstellen eines Textfelds für AutoText

Wenn ein Textfeld `AutoText` unterstützt und Benutzer zweimal die Leertaste drücken, fügt das BlackBerry®-Gerät einen Punkt ein, schreibt den nächsten Buchstaben nach einem Punkt groß und ersetzt Wörter gemäß den Definitionen in der `AutoText`-Anwendung.

1. Importieren Sie die folgenden Klassen:
 - `net.rim.device.api.ui.component.AutoTextEditField`
 - `net.rim.device.api.ui.autotext.AutoText`
 - `net.rim.device.api.ui.component.BasicEditField`
2. Erstellen Sie eine Instanz von `AutoTextEditField`.

```
AutoTextEditField autoT = new AutoTextEditField("AutoTextEditField: ", "");
```

Erstellen eines Datumsfelds

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.lang.*;
```

- Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyUiScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyUi extends UiApplication
{
    public static void main(String[] args)
    {
        MyUi theApp = new MyUi();
        theApp.enterEventDispatcher();
    }
    public MyUi()
    {
        pushScreen(new MyUiScreen());
    }
}
```

- Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyUiScreen extends MainScreen
{
    public MyUiScreen()
    {
        setTitle("UI Component Sample");
    }
}
```

- Erstellen Sie im Bildschirmkonstruktor mithilfe der `DateField`-Klasse ein Datumsfeld. Stellen Sie `System.currentTimeMillis()` als Parameter bereit, um die aktuelle Uhrzeit zurückzugeben. Verwenden Sie den `DateField.DATE_TIME`-Stil, um sowohl das Datum als auch die Uhrzeit anzuzeigen. Sie können andere Stile verwenden, um nur das Datum oder nur die Uhrzeit anzuzeigen.

```
add(new DateField("Date: ", System.currentTimeMillis(),
    DateField.DATE_TIME));
```

Erstellen eines Kennwortfelds

- Importieren Sie die `net.rim.device.api.ui.component.PasswordEditField`-Klasse.
- Erstellen Sie eine Instanz von `PasswordEditField`.

Die folgende Instanz verwendet zum Beispiel einen Konstruktor, mit dem Sie einen Standardanfangswert für `PasswordEditField` bereitstellen können:

```
PasswordEditField pwd = new PasswordEditField("PasswordEditField: ", "");
```

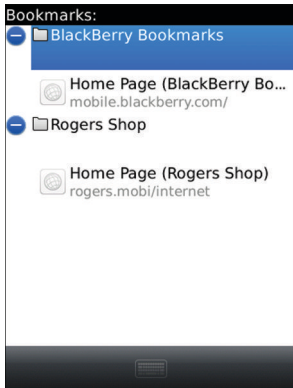
Strukturansichten

Verwenden Sie eine Strukturansicht, um Objekte (z. B. Ordner) in einer hierarchischen Weise anzuzeigen.

Objekte in der Strukturansicht sind Knoten. Der oberste Knoten ist der Stammknoten. Ein Knoten im Baum kann über untergeordnete Knoten verfügen. Ein Knoten, der über einen untergeordneten Knoten verfügt, ist ein übergeordneter Knoten.

Benutzer können in einer Strukturansicht die folgende Aktion ausführen:

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Erweitern oder Reduzieren eines Objekts mit einem Pluszeichen (+) oder Minuszeichen (-) in einer Hierarchie.	Drücken der Leertaste oder Klicken mit dem Trackpad.	• Tippen auf das Objekt. • Drücken Sie die Leertaste. • Klicken mit dem Trackpad.



Bewährtes Verfahren: Implementieren von Strukturansichten

- Verwenden Sie die `TreeField`-Klasse, um Strukturansichten zu erstellen.
- Stellen Sie ein Popup-Menü bereit, wenn Benutzer beim Anklicken eines übergeordneten Knotens mehrere Aktionen ausführen können.
- Nehmen Sie nur dann einen Stammknoten auf, wenn die Benutzer Aktionen für die gesamte Struktur ausführen können müssen. Verzichten Sie anderenfalls auf den Stammknoten.

Erstellen eines Feldes für das Anzeigen einer Strukturansicht

Verwenden Sie eine Strukturansicht, um Objekte, wie z. B. eine Ordnerstruktur, auf hierarchische Weise anzuzeigen. `TreeField` enthält Knoten. Der höchste Knoten ist der Stammknoten. Ein Knoten in der Struktur kann untergeordnete Knoten besitzen. Ein Knoten, der ein untergeordnetes Element besitzt, ist ein übergeordneter Knoten.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.component.TreeField;
import net.rim.device.api.ui.component.TreeFieldCallback;
import net.rim.device.api.ui.container.MainScreen;
import java.lang.String;
```

2. Implementieren Sie die `TreeFieldCallback`-Schnittstelle.
3. Rufen Sie `TreeField.setExpanded()` für das `TreeField`-Objekt auf, um anzugeben, ob ein Ordner ausblendbar ist. Erstellen Sie ein `TreeField`-Objekt und mehrere untergeordnete Knoten für das `TreeField`-Objekt. Rufen Sie `TreeField.setExpanded()` mithilfe von `node4` als Parameter auf, um den Ordner auszublenden.

```
String fieldOne = new String("Main folder");
...
TreeCallback myCallback = new TreeCallback();
TreeField myTree = new TreeField(myCallback, Field.FOCUSABLE);
int node1 = myTree.addChildNode(0, fieldOne);
int node2 = myTree.addChildNode(0, fieldTwo);
int node3 = myTree.addChildNode(node2, fieldThree);
int node4 = myTree.addChildNode(node3, fieldFour);
...
int node10 = myTree.addChildNode(node1, fieldTen);
myTree.setExpanded(node4, false);
...
mainScreen.add(myTree);
```

4. Um ein `TreeField`-Objekt neu zu zeichnen, wenn sich ein Knoten ändert, erstellen Sie eine Klasse, die die `TreeFieldCallback`-Schnittstelle implementiert, und implementieren Sie die `TreeFieldCallback.drawTreeItem`-Methode. Die `TreeFieldCallback.drawTreeItem`-Methode verwendet das Cookie für einen Strukturknoten, um an der Position eines Knotens ein `String`-Objekt zu zeichnen. Die `TreeFieldCallback.drawTreeItem`-Methode ruft `Graphics.drawText()` auf, um `String` zu zeichnen.

```
private class TreeCallback implements TreeFieldCallback
{
    public void drawTreeItem(TreeField _tree, Graphics g, int node, int y, int
width, int indent)
    {
        String text = (String)_tree.getCookie(node);
        g.drawText(text, indent, y);
    }
}
```

Bilder

9

Verwenden von codierten Bildern

Zugriff auf ein codiertes Bild über einen Eingabestream

1. Importieren Sie die erforderlichen Klassen.

```
import java.io.InputStream;
```

2. Speichern Sie das Bild im Projektordner oder in einem Unterordner.
3. Fügen Sie das Bild dem Projekt im BlackBerry® Java® Plug-in Eclipse® oder im BlackBerry® Java® Development Environment hinzu.
4. Rufen Sie `getClass().getResourceAsStream()` auf, um das Bild als Eingabestream in Bytes abzurufen.

```
private InputStream input;  
  
Class _class = this.getClass();  
input = _class.getResourceAsStream("/images/example.png");
```

Codieren eines Bilds

1. Importieren Sie die erforderlichen Klassen.

```
import net.rim.device.api.system.EncodedImage;
```

2. Rufen Sie `EncodedImage.createEncodedImage()` auf. Diese Methode verwendet die unverarbeiteten Bilddaten im Byte-Array, um eine `EncodedImage`-Instanz zu erstellen.
3. Prüfen Sie das `IllegalArgumentException`-Objekt, das von `EncodedImage.createEncodedImage()` ausgegeben wird, wenn das von Ihnen als Parameter bereitgestellte Byte-Array kein anerkanntes Bildformat enthält.

```
// Store the contents of the image file.  
private byte[] data = new byte[2430];  
try {  
    // Read the image data into the byte array  
    input.read(data);  
} catch (IOException e) {  
    // Handle exception.  
}  
try {  
    EncodedImage image = EncodedImage.createEncodedImage(data, 0, data.length);  
} catch (IllegalArgumentException iae) {  
    System.out.println("Image format not recognized.");  
}
```

Anzeigen eines codierten Bilds

1. Importieren Sie die erforderliche Klasse.

```
import net.rim.device.api.ui.component.BitmapField;
```

2. Rufen Sie `BitmapField.setImage()` auf, um das codierte Bild einem `BitmapField` zuzuweisen.
3. Rufen Sie `add()` auf, um dem Bildschirm das `BitmapField` hinzuzufügen.

```
BitmapField field = new BitmapField();  
field.setImage(image);  
add(field);
```

Festlegen der Anzeigegröße eines codierten Bilds

1. Importieren Sie die erforderliche Klasse.

```
import net.rim.device.api.system.EncodedImage;
```

2. Rufen Sie `EncodedImage.scaleImage32(int scaleX, int scaleY)` auf. Die übergebenen Skalierungswert-Parameter müssen `net.rim.device.api.math.Fixed32`-Zahlen sein ($0 < \text{Skalierung} < 1$ für das Hochskalieren und $\text{Skalierung} > 1$ für das Herunterskalieren). `scaleImage32()` erstellt ein neues `EncodedImage`, statt das aktuelle Bild zu ändern.

Festlegen des Decodierungsmodus für ein Bild

1. Importieren Sie die erforderliche Klasse.

```
import net.rim.device.api.system.EncodedImage;
```

2. Rufen Sie `EncodedImage.setDecodeMode()` auf und verwenden Sie dabei einen der folgenden Modi als Parameter:
 - Verwenden Sie `DECODE_ALPHA`, um einen Alphakanal zu decodieren, sofern vorhanden (dies ist der Standardmodus).
 - Verwenden Sie `DECODE_NATIVE`, um die Decodierung eines Bitmap-Bilds in den systemeigenen Bitmap-Bildtyp der BlackBerry® Device Software zu erzwingen.
 - Verwenden Sie `DECODE_READONLY`, um das decodierte Bitmap-Bild als schreibgeschützt zu markieren.

Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken

Die `ZoomScreen`-Klasse ermöglicht Ihnen die Bereitstellung der Bildfunktionalitäten Schwenken und Vergrößern/Verkleinern. Wenn ein BlackBerry®-Gerätebenutzer mit dem Trackball oder auf den Touchscreen klickt, zeigt der Bildschirm den mittleren Bereich des vergrößerten Bilds an.

Wenn das Bild vergrößert wird, wird auf dem Bildschirm zudem ein Overlay angezeigt, das den vergrößerten Bereich hervorhebt. Durch Scrollen mit dem Trackball oder Bewegen eines Fingers auf dem Bildschirm wird das Bild geschwenkt. Wenn der Benutzer das Vergrößern und Schwenken beendet, wird das Overlay auf dem Bildschirm ausgeblendet.

Auf BlackBerry-Geräten mit Touchscreen können Benutzer den Bereich des zu vergrößernden Bilds definieren. Der Benutzer kann zwei Punkte auf dem Bild berühren, um die diagonal entgegengesetzten Ecken des Bereichs zu definieren. Nach der Auswahl des Bereichs kann der Benutzer einen Finger über das Bild bewegen, um den zu vergrößernden Bereich zu verschieben. Um den definierten Bereich zu vergrößern, klickt der Benutzer auf den Bildschirm. Zum Verkleinern drückt der Benutzer die Esc-Taste.

Codebeispiel: Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.extension.container.*;
public class ZoomableImageApp extends UiApplication
{
    public static void main(String[] args)
    {
        ZoomableImageApp theApp = new ZoomableImageApp();
        theApp.enterEventDispatcher();
    }
    public ZoomableImageApp()
    {
        EncodedImage myImg = new EncodedImage("myImg.jpg");
        ZoomScreen zoomableImg = new ZoomScreen(myImg);
        pushScreen(zoomableImg);
    }
}
```

Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken

1. Importieren Sie die erforderlichen Klassen und Schnittstellen:

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.extension.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Erstellen Sie im Konstruktor mithilfe einer Ressource im Projekt ein `EncodedImage`, erstellen Sie mit dem `EncodedImage` einen `ZoomScreen` und rufen Sie `pushScreen()` auf, um das Bild zum Vergrößern und Schwenken anzuzeigen.

```
public class ZoomableImageApp extends UiApplication
{
    public static void main(String[] args)
    {
        ZoomableImageApp theApp = new ZoomableImageApp();
        theApp.enterEventDispatcher();
    }
    public ZoomableImageApp()
    {
        EncodedImage myImg = new EncodedImage("myImg.jpg");
        ZoomScreen zoomableImg = new ZoomScreen(myImg);
        pushScreen(zoomableImg);
    }
}
```

Anzeigen mehrerer Bilder zum Scrollen

Sie können die `PictureScrollField`-Klasse verwenden, um eine horizontale Zeile mit Bildern wiederzugeben, durch die der Benutzer mithilfe des Trackballs oder durch Berühren scrollen kann.

Mit dem `PictureScrollField` können Sie den Markierungsstil des ausgewählten Bilds, den Hintergrundstil vom `PictureScrollField`, den Anzeigetext unter dem ausgewählten Bild, den beim ersten Auswählen eines Bilds angezeigten `QuickInfo`-Text sowie die Höhe und die Breite der Bilder definieren.

Codebeispiel: Anzeigen einer Reihe von Bildern zum Scrollen

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.ui.extension.component.*;
public class PictureScrollFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        PictureScrollFieldDemo app = new PictureScrollFieldDemo();
        app.enterEventDispatcher();
    }
    public PictureScrollFieldDemo()
    {
        pushScreen(new PictureScrollFieldDemoScreen());
    }
}
class PictureScrollFieldDemoScreen extends MainScreen
{
    public PictureScrollFieldDemoScreen()
    {
        setTitle("PictureScrollField Demo");
        Bitmap[] images = new Bitmap[3];
        String[] labels = new String[3];
        String[] tooltips = new String[3];
```

```

        images[0] = Bitmap.getBitmapResource("img1.jpg");
        labels[0] = "Label for image 1";
        tooltips[0] = "Tooltip for image 1";
        images[1] = Bitmap.getBitmapResource("img2.jpg");
        labels[1] = "Label for image 2";
        tooltips[1] = "Tooltip for image 2";
        images[2] = Bitmap.getBitmapResource("img3.jpg");
        labels[2] = "Label for image 3";
        tooltips[2] = "Tooltip for image 3";
        ScrollEntry[] entries = new ScrollEntry[3];
        for (int i = 0; i < entries.length; i++)
        {
            entries[i] = new ScrollEntry(images[i], labels[i], tooltips[i]);
        }
        PictureScrollField pictureScrollField = new PictureScrollField(150, 100);
        pictureScrollField.setData(entries, 0);
        pictureScrollField.setHighlightStyle(HighlightStyle.ILLUMINATE);
        pictureScrollField.setHighlightBorderColor(Color.BLUE);
        pictureScrollField.setBackground
(BackgroundFactory.createSolidTransparentBackground(Color.RED, 150));
        pictureScrollField.setLabelsVisible(true);
        add(pictureScrollField);
    }
}

```

Anzeigen mehrerer Bilder zum Scrollen

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.ui.extension.component.*;

```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie im Konstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `PictureScrollFieldDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```

public class PictureScrollFieldDemo extends UiApplication
{
    public static void main(String[] args)
    {
        PictureScrollFieldDemo app = new PictureScrollFieldDemo();
        app.enterEventDispatcher();
    }
    public PictureScrollFieldDemo()
    {
        pushScreen(new PictureScrollFieldDemoScreen());
    }
}

```

3. Erstellen Sie das Framework für den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse.

```
class PictureScrollFieldDemoScreen extends MainScreen
{
    public PictureScrollFieldDemoScreen()
    {
    }
}
```

4. Rufen Sie im Konstruktor `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird.

```
setTitle("PictureScrollField Demo");
```

5. Erstellen und initialisieren Sie im Konstruktor drei Arrays, um die im `PictureScrollField` anzuzeigenden Bilder sowie die Bezeichnungen und den `QuickInfo`-Text, der beim Auswählen der einzelnen Bilder angezeigt wird, zu speichern. In diesem Beispiel enthält das `PictureScrollField` drei Bilder.

```
Bitmap[] images = new Bitmap[3];
String[] labels = new String[3];
String[] tooltips = new String[3];
images[0] = Bitmap.getBitmapResource("img1.jpg");
labels[0] = "Label for image 1";
tooltips[0] = "Tooltip for image 1";
images[1] = Bitmap.getBitmapResource("img2.jpg");
labels[1] = "Label for image 2";
tooltips[1] = "Tooltip for image 2";
images[2] = Bitmap.getBitmapResource("img3.jpg");
labels[2] = "Label for image 3";
tooltips[2] = "Tooltip for image 3";
```

6. Erstellen und initialisieren Sie im Konstruktor ein Array mit `ScrollEntry`-Objekten. Ein `ScrollEntry` stellt jedes Element im `PictureScrollField` dar.

```
ScrollEntry[] entries = new ScrollEntry[3];
for (int i = 0; i < entries.length; i++)
{
    entries[i] = new ScrollEntry(images[i], labels[i], tooltips[i]);
}
```

7. Erstellen und initialisieren Sie im Konstruktor das `PictureScrollField` und verwenden Sie für jedes Bild eine Breite von 150 Pixeln und eine Höhe von 100 Pixeln. Rufen Sie `setData()` auf, um die Einträge im `PictureScrollField` festzulegen. Der zweite Parameter in `setData()` legt fest, welche Bildposition standardmäßig ausgewählt wird.

```
PictureScrollField pictureScrollField = new PictureScrollField(150,
    100);
pictureScrollField.setData(entries, 0);
```

8. Legen Sie im Konstruktor den `PictureScrollField`-Stil fest. Rufen Sie `setHighlightStyle()` auf, um festzulegen, wie das ausgewählte Bild hervorgehoben wird. Rufen Sie `setHighlightBorderColor()` auf, um die Rahmenfarbe des ausgewählten Bilds festzulegen. Rufen Sie `setBackground()` auf, um den `PictureScrollField`-Hintergrund festzulegen. Rufen Sie `setLabelVisible()` auf, um festzulegen, ob das `PictureScrollField` die Bezeichnung des ausgewählten Bilds anzeigt.

```
pictureScrollField.setHighlightStyle(HighlightStyle.ILLUMINATE);  
pictureScrollField.setHighlightBorderColor(Color.BLUE);  
pictureScrollField.setBackground(BackgroundFactory  
    .createSolidTransparentBackground(Color.RED, 150));  
pictureScrollField.setLabelsVisible(true);
```

9. Rufen Sie im Konstruktor `add()` auf, um das `PictureScrollField` anzuzeigen.

```
add(pictureScrollField);
```

Menübefehle

10

Erstellen eines Menüs

Die `MainScreen`-Klasse stellt Standardkomponenten einer BlackBerry®-Geräteanwendung bereit. Sie enthält ein Standardmenü.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `CreateMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class CreateMenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateMenu theApp = new CreateMenu();
        theApp.enterEventDispatcher();
    }
    public CreateMenu()
    {
        pushScreen(new CreateMenuScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um ein Textfeld zum Bildschirm hinzuzufügen. Rufen Sie `addMenuItem()` auf, um einen Menüeintrag zu dem Menü hinzuzufügen, das `MainScreen` erstellt.

```
class CreateMenuScreen extends MainScreen
{
    public CreateMenuScreen()
    {
        setTitle("Create Menu Sample");
        add(new RichTextField("Create a menu"));
        addMenuItem(_viewItem);
    }
}
```

4. Erstellen Sie den Menüeintrag mithilfe der `MenuItem`-Klasse. Überschreiben Sie `run()`, um die Aktion anzugeben, die ausgeführt wird, wenn der Benutzer auf den Menüeintrag klickt. Wenn der Benutzer auf den Menüeintrag klickt, ruft die Anwendung `Menu.run()` auf.

```
private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
{
    public void run()
    {
        Dialog.inform("Display more information");
    }
};
```

5. Überschreiben Sie `close()`, um ein Dialogfeld anzuzeigen, wenn der Benutzer auf den Menüeintrag "Schließen" klickt. Standardmäßig ist der Menüeintrag "Schließen" in dem Menü enthalten, das `MainScreen` erstellt. Rufen Sie `super.close()` auf, um die Anwendung zu schließen. Wenn der Benutzer das Dialogfeld schließt, ruft die Anwendung `MainScreen.close()` auf, um die Anwendung zu schließen.

```
public void close()
{
    Dialog.alert("Goodbye!");
    super.close();
}
```

Codebeispiel: Erstellen eines Menüs

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class CreateMenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateMenu theApp = new CreateMenu();
        theApp.enterEventDispatcher();
    }
    public CreateMenu()
    {
        pushScreen(new CreateMenuScreen());
    }
}
class CreateMenuScreen extends MainScreen
{
    public CreateMenuScreen()
    {
        setTitle("Create Menu Sample");
        add(new RichTextField("Create a menu"));
        addMenuItem(_viewItem);
    }
    private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
    {
        public void run()
        {
            Dialog.inform("Display more information");
        }
    };
    public void close()
    {
        Dialog.alert("Goodbye!");
    }
}
```

```
        super.close();  
    }  
}
```

Bewährtes Verfahren: Implementieren von Menüs

- Stellen Sie stets ein vollständiges Menü bereit.
- Stellen Sie sicher, dass Benutzer die Menütaste drücken können, um das vollständige Menü zu öffnen, und eine Aktion zu initiieren, wenn ein Menüelement markiert ist. Stellen Sie sicher, dass Benutzer auch die Menütaste halten können, um das Dialogfeld zum Wechseln von Anwendungen zu öffnen.
- Verwenden Sie als Standardmenüelement das Menüelement, das die Benutzer am wahrscheinlichsten auswählen werden.
- Platzieren Sie den Standardmenüeintrag und die anderen am häufigsten verwendeten Menüeinträge in der Mitte des Menüs.
- Vergewissern Sie sich, dass die Reihenfolge der Menüeinträge der Reihenfolge der Menüeinträge in anderen BlackBerry®-Geräteanwendungen entspricht.
- Gruppieren Sie Menüeinträge entsprechend der Verwendungsweise oder der Funktionalität und lassen Sie Ihre Gruppierungen wenn möglich von den Benutzern testen.
- Fügen Sie Trennzeichen zwischen den Menüeintragsgruppen ein.
- Platzieren Sie keine Menüeinträge für einander widersprechende Aktionen nahe beieinander. Platzieren Sie zum Beispiel nicht den Menüeintrag "Löschen" neben dem Menüeintrag "Öffnen".
- Stellen Sie stets die Menüeinträge "Anwendung wechseln" und "Schließen" bereit. Platzieren Sie diese Menüeinträge am Ende des Menüs. Wenn Sie Standardkomponenten verwenden, sind diese Menüeinträge automatisch enthalten.

Richtlinien für Bezeichnungen

- Verwenden Sie kurze und klare beschreibende Bezeichnungen, die nicht länger als 12 Zeichen sind. Wenn eine Bezeichnung zu lang ist, werden Auslassungspunkte (...) angezeigt, um zu verdeutlichen, dass der Text abgeschnitten ist.
- Verwenden Sie Verben für Bezeichnungen.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Bezeichnungen.
- Verwenden Sie Auslassungspunkte in einer Menüeintragsbezeichnung, um anzuzeigen, dass Benutzer nach dem Klicken auf den Menüeintrag eine weitere Aktion ausführen müssen. Wenn Benutzer zum Beispiel im Kalender auf den Menüeintrag "Wechseln zu Datum..." klicken, müssen sie auf dem eingeblendeten Bildschirm ein Datum angeben.
- Vermeiden Sie in Bezeichnungen die Verwendung von Symbolen wie z. B. Sternchen (*).

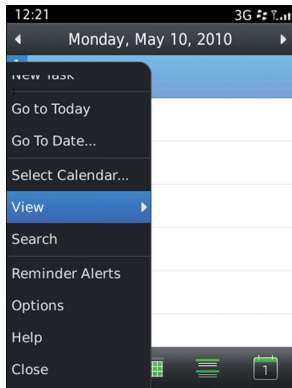
Untermenüs

Ein Untermenü besteht aus einer Gruppe von verwandten Menüeinträgen, die als Teilmenge eines im vollständigen Menü vorhandenen Menüeintrags angezeigt werden. Wenn Sie Elemente in ein Untermenü aufnehmen, können Benutzer häufig verwendete oder wichtige Elemente problemloser im vollständigen Menü finden. Untermenüs enthalten in der Regel die folgenden Arten von Aktionen:

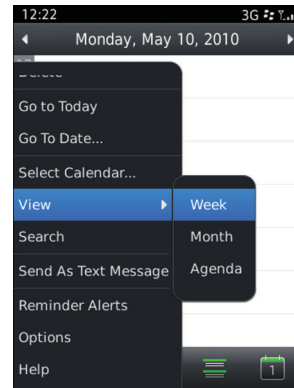
- Senden von Elementen auf verschiedenen Wegen (zum Beispiel das Versenden eines Bilds in einer E-Mail-Nachricht, einer MMS-Nachricht oder als Audiopostkarte)
- Sortieren, Suchen, Finden und Filtern von Elementen auf unterschiedliche Weisen (zum Beispiel das Filtern von Nachrichten nach Absender oder Betreff)
- Ändern von Ansichten (zum Beispiel die Tages-, Wochen- oder Tagesordnungsansicht in einem Kalender)

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Ein Untermenü für einen Menüeintrag anzeigen, der im vollständigen Menü markiert ist.	• Klicken mit dem Trackpad. • Bewegen Sie einen Finger nach rechts über das Trackpad. • Drücken Sie die Taste Menü. • Drücken Sie die Eingabetaste.	• Tippen Sie auf den Menüeintrag. • Klicken mit dem Trackpad. • Bewegen Sie einen Finger nach rechts über das Trackpad. • Drücken Sie die Taste Menü. • Drücken Sie die Eingabetaste.
Menüeintrag aus einem Untermenü auswählen.	• Klicken mit dem Trackpad. • Drücken Sie die Taste Menü. • Drücken Sie die Eingabetaste.	• Tippen Sie auf den Menüeintrag. • Klicken mit dem Trackpad. • Drücken Sie die Taste Menü. • Drücken Sie die Eingabetaste.
Untermenü schließen.	• Bewegen Sie einen Finger nach links über das Trackpad. • Drücken der Esc-Taste	• Tippen Sie auf eine Position außerhalb des Untermenüs. • Bewegen Sie einen Finger nach links über das Trackpad. • Drücken der Esc-Taste
Untermenü und vollständiges Menü schließen.	Drücken Sie zweimal die ESC-Taste.	• Tippen Sie zweimal auf eine Position außerhalb des vollständigen Menüs und des Untermenüs. • Drücken Sie zweimal die ESC-Taste. • Öffnen oder Schließen der Schiebehülle

Ein Pfeil zeigt an, dass Untermenüeinträge für ein Element des vollständigen Menüs verfügbar sind.



Ein Element im vollständigen Menü mit einem Pfeil



Ein vollständiges Menü und ein Untermenü

Bewährtes Verfahren: Implementieren von Untermenüs

- Verwenden Sie die Submenu-Unterklasse, um Untermenüs zu erstellen.
- Verwenden Sie Untermenüs, um im vollständigen Menü die Anzahl der Menüeinträge zu verringern. Wenn z. B. die Benutzer scrollen müssen, um in einem vollständigen Menü alle Elemente anzuzeigen, ordnen Sie einige Menüeinträge in einem Untermenü an.
- Ordnen Sie verwandte Elemente in einem Untermenü an. Wenn im vollständigen Menü z. B. "Sortieren nach" angezeigt wird, ordnen Sie "Datum", "Name" und "Betreff" in einem Untermenü an.
- Erweiterte Funktionen sollten in einem Untermenü angeordnet werden. Verwenden Sie z. B. "Zusätzliche Optionen" im vollständigen Menü und ordnen Sie die Optionen im Untermenü an.
- Vermeiden Sie die Verwendung von Untermenüs für einen oder zwei Menüeinträge.
- Wenn ein Untermenü mehr als sechs Elemente enthält, sollten Sie die Elemente im Untermenü in zwei Abschnitten anordnen. Platzieren Sie die häufig verwendeten Menüeinträge oben im Untermenü, fügen Sie ein Trennzeichen ein und ordnen Sie anschließend die übrigen Elemente alphabetisch an.
- Wenn ein Menüeintrag mehr als nur ein Klick vom Benutzer erfordert (z. B. wenn der Benutzer ein Datum festlegen muss), öffnen Sie ein Dialogfeld.
- Vermeiden Sie, dass häufig verwendete oder wichtige Menüeinträge in Untermenüs angeordnet werden.
- Vermeiden Sie verschachtelte Untermenüs.

Richtlinien für Bezeichnungen

- Verwenden Sie im vollständigen Menü kurze, aussagekräftige Bezeichnungen, die die Aktion deutlich definieren. Benutzer sollten nicht das Untermenü öffnen müssen, um die Elementbedeutung im vollständigen Menü zu verstehen.
- Verwenden Sie im vollständigen Menü verbale Bezeichnungen. Verwenden Sie Substantive nur dann, wenn die Bedeutung klar ist. Wird z. B. "E-Mail-Konten" im vollständigen Menü angezeigt, ordnen Sie jedes E-Mail-Konto in einem Untermenü an.

- Vermeiden Sie, dass sich Verben in Untermenüs wiederholen. Vermeiden Sie zum Beispiel die Verwendung von "Sortieren nach" im vollständigen Menü und "Sortieren nach Datum" sowie "Sortieren nach Namen" im Untermenü.
- Verwenden Sie die übliche Groß-/Kleinschreibung für Bezeichnungen. Schreiben Sie das erste Wort im Untermenü auch dann groß, wenn es eine Präposition ist. Verwenden Sie zum Beispiel "Senden" > "Als E-Mail" und nicht "als E-Mail".
- Vermeiden Sie den Begriff "Mehr" im vollständigen Menü, es sei denn, dass die Position des Menüeintrags verdeutlicht, auf was sich "Mehr" bezieht. Werden z. B. im vollständigen Menü die Einträge "Posteingang", "Postausgang" und "Mehr" zwischen Trennzeichen angezeigt, ordnen Sie die mit einer Mailbox verknüpften zusätzlichen Optionen in einem Untermenü an.

Erstellen eines Untermenüs

Sie können ein Untermenü für Ihre BlackBerry®-Geräteanwendung erstellen. Ein Untermenü wird angezeigt, wenn ein Benutzer einen Menüeintrag auswählt, der mit einem Untermenü verknüpft ist.

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `CreateSubmenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class CreateSubmenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateSubmenu theApp = new CreateSubmenu();
        theApp.enterEventDispatcher();
    }
    public CreateSubmenu()
    {
        pushScreen(new CreateSubmenuScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um ein Textfeld zum Bildschirm hinzuzufügen.

```
class CreateSubmenuScreen extends MainScreen
{
    public CreateSubmenuScreen()
    {
        setTitle("Create Submenu Sample");
    }
}
```

```

        add(new RichTextField("Create a submenu"));
    }
}

```

4. Erstellen Sie mithilfe der MenuItem-Klasse die Elemente des Untermenüs. Überschreiben Sie für jedes Element im Untermenü `run()`, um die Aktion anzugeben, die ausgeführt wird, wenn der Benutzer auf den Menüeintrag klickt. Wenn der Benutzer auf den Menüeintrag klickt, ruft die Anwendung `Menu.run()` auf.

```

private MenuItem _status1 = new MenuItem("Available", 100, 1)
{
    public void run()
    {
        Dialog.inform("I'm available");
    }
};
private MenuItem _status2 = new MenuItem("Unavailable", 200, 2)
{
    public void run()
    {
        Dialog.inform("I'm unavailable");
    }
};

```

5. Überschreiben Sie `makeMenu()`, um das Menü für die Anwendung zu erstellen. Erstellen Sie das Untermenü mithilfe der SubMenu-Klasse. Rufen Sie `add()` auf, um dem Untermenü die Elemente des Untermenüs hinzuzufügen. Rufen Sie `super.makeMenu()` auf, um das Menü zu erstellen.

```

protected void makeMenu( Menu menu, int instance )
{
    SubMenu statusSubMenu = new SubMenu(null,"My Status",300,3);
    statusSubMenu.add(_status1);
    statusSubMenu.add(_status2);
    menu.add(statusSubMenu);
    super.makeMenu(menu, instance);
};

```

Codebeispiel: Erstellen eines Untermenüs

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
public class CreateSubmenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateSubmenu theApp = new CreateSubmenu();
        theApp.enterEventDispatcher();
    }
    public CreateSubmenu()
    {
        pushScreen(new CreateSubmenuScreen());
    }
}
class CreateSubmenuScreen extends MainScreen

```

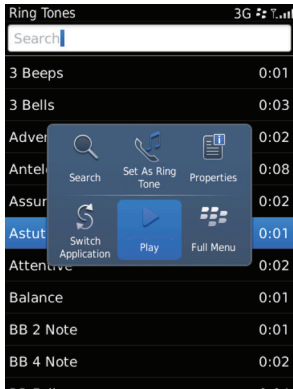
```
{
    public CreateSubmenuScreen()
    {
        setTitle("Create Submenu Sample");
        add(new RichTextField("Create a submenu"));
    }
    private MenuItem _status1 = new MenuItem("Available", 100, 1)
    {
        public void run()
        {
            Dialog.inform("I'm available");
        }
    };
    private MenuItem _status2 = new MenuItem("Unavailable", 200, 2)
    {
        public void run()
        {
            Dialog.inform("I'm unavailable");
        }
    };
    protected void makeMenu( Menu menu, int instance )
    {
        SubMenu statusSubMenu = new SubMenu(null,"My Status",300,3);
        statusSubMenu.add(_status1);
        statusSubMenu.add(_status2);
        menu.add(statusSubMenu);
        super.makeMenu(menu, instance);
    };
}
```

Popup-Menüs

Ein Popup-Menü bietet den Benutzern eine schnelle Möglichkeit, um auf die häufigsten Aktionen für ein markiertes Element zuzugreifen. Sie können ein Popup-Menü auch dann verwenden, wenn ein markiertes Element mit mehreren Aktionen verknüpft ist oder wenn mit einem markierten Element keine primären Aktionen verknüpft sind. Sie können ein Popup-Menü erstellen, das neun Aktionen, sechs Aktionen oder drei Aktionen enthält.

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
Öffnen eines Popup-Menü.	• Klicken mit dem Trackpad. • Wenn bereits eine primäre Aktion mit einem Element verknüpft wurde (zum Beispiel das Öffnen einer Nachricht), können die Benutzer mit dem Trackpad klicken und es halten, um ein Popup-Menü zu öffnen.	• Tippen auf das Element • Klicken mit dem Trackpad. • Wenn bereits eine primäre Aktion mit einem Element verknüpft wurde (zum Beispiel das Öffnen einer Nachricht), können die Benutzer mit dem Trackpad klicken und es halten oder den

Aktion	BlackBerry-Geräte nur mit Trackpad	BlackBerry-Geräte mit Touchscreen und Trackpad
		Touchscreen mit einem Finger berühren, um ein Popup-Menü zu öffnen.
Auswählen eines Elements aus einem Popup-Menü.	• Klicken mit dem Trackpad. • Drücken Sie die Eingabetaste.	• Tippen auf das Element • Klicken mit dem Trackpad. • Drücken Sie die Eingabetaste.
Schließen eines Popup-Menüs.	Drücken der Esc-Taste	• Tippen Sie auf eine Position außerhalb des Popup-Menüs. • Drücken der Esc-Taste • Öffnen oder Schließen der Schiebehülle



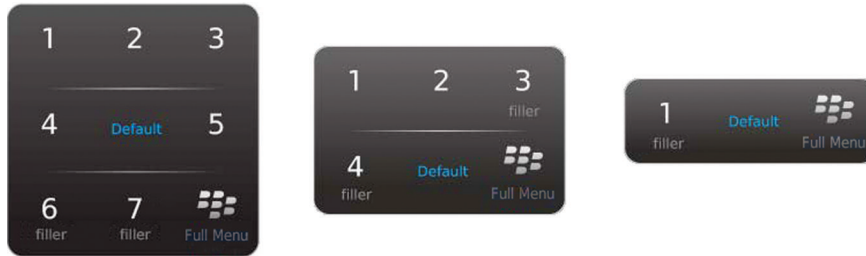
Popup-Menüs ersetzen Kontextmenüs oder kurze Menüs. Alle vorhandenen Kontextmenüs werden automatisch in Popup-Menüs konvertiert.

Bewährtes Verfahren: Implementieren von Popup-Menüs

- Verwenden Sie Popup-Menüs statt Kontextmenüs (kurzer Menüs). Stellen Sie sicher, dass das Popup-Menü für den Benutzer nutzbringend ist.
- Legen Sie den Menüeintrag, der von den Benutzern am wahrscheinlichsten gewählt wird, als Standardmenüeintrag fest. Der Standardeintrag im Popup-Menü sollte mit dem Standardeintrag im vollständigen Menü identisch sein.
- Stellen Sie nur die häufigsten Aktionen für ein markiertes Element in einem Popup-Menü bereit.
- Stellen Sie ein Symbol und eine Bezeichnung für jedes Element im Popup-Menü bereit. Erstellen Sie Symbole mit einer durchschnittlichen Größe von 33x33 Pixel. Diese Symbole werden in einem 60x40 Pixelbereich angezeigt und sollten negative Abstandsorientierungen enthalten.

Richtlinien für das Einfügen von Elementen in Popup-Menüs

- Platzieren Sie den Standardmenüeintrag in die Mitte des Popup-Menüs.
- Ordnen Sie die übrigen Elemente entsprechend den unten stehenden nummerierten Positionen von am häufigsten bis zu den am wenigsten verwendeten Elementen ein. Nutzen Sie das Gedächtnisvermögen der Benutzer und versuchen Sie, die Reihenfolge der Aktionen verträglich zur Aktionenreihenfolge in anderen BlackBerry®-Geräteanwendungen zu implementieren.



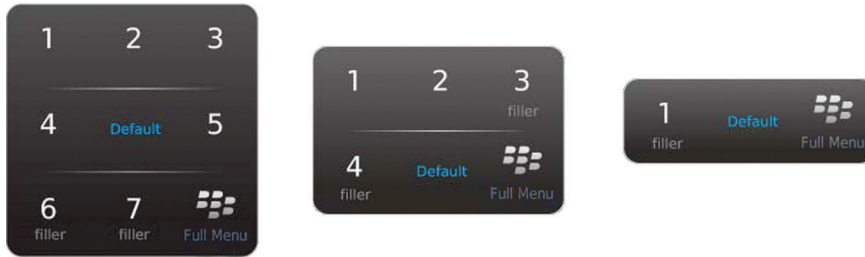
- Wenn die Positionen dynamisch gefüllt werden, zeigen Sie keine Menüeinträge an, die nicht verfügbar sind. Erlauben Sie, dass sich die Position der verfügbaren Menüeinträge ändern kann.
- Wenn die Anzahl der Aktionen zum Füllen eines Menüs nicht ausreicht, verwenden Sie ein kleineres Menü. Wenn eine oder zwei Positionen aufgefüllt werden müssen, verwenden Sie nützliche Aktionen wie z. B. "Kopieren" oder "Suchen". Wenn Sie eine Position nicht füllen, wird "Anwendung wechseln" oder "Startseite" angezeigt.

Info über das Einfügen von Elementen in Popup-Menüs

Popup-Menüs zeigen in Abhängigkeit von der Elementanzahl Menüeinträge in einem Rasterformat 3x3, 3x2 oder 3x1 an. Standardmäßig enthält das Popup-Menü immer ein Element zum Öffnen des vollständigen Menüs. Sie können bis zu acht zusätzliche Elemente bereitstellen. Wenn Sie mehr als acht Elemente bereitstellen, werden die zusätzlichen Elemente nicht angezeigt. Stattdessen können Sie dem vollständigen Menü zusätzliche Elemente hinzufügen.

Das erste Element, das Sie dem Vektor der `MenuItem`-Elemente hinzufügen, ist das Standardelement im Popup-Menü. Das Standardelement wird in der Mitte des Rasters hervorgehoben angezeigt, wenn der BlackBerry®-Gerätebenutzer das Popup-Menü öffnet. Die anderen Elemente werden im Popup-Menü in der Reihenfolge positioniert, in der Sie sie hinzufügen.

Wenn die Anzahl der im Popup-Menü hinzugefügten Elemente Zellen im Raster frei lassen, werden dem Menü Füllelemente hinzugefügt. Das erste Füllelement ist eine Option zum Wechseln von Anwendungen. Das zweite Füllelement ist eine Option zum Zurückkehren zur Startseite.



Unterstützung für ältere Kontextmenüs

Seit BlackBerry® Java® SDK 6.0 wird das Kontextmenü oder das kurze Menü einer Anwendung in ein Popup-Menü umgewandelt. Unabhängig davon, ob Sie Menüeinträge zu einem Kontextmenü hinzugefügt oder die einem Kontextmenü standardmäßig zugefügten Elemente verwendet haben, muss Ihr Code nicht geändert werden, um diese Elemente stattdessen in einem Popup-Menü anzuzeigen. BlackBerry Java SDK 6.0 enthält jedoch Klassen und Schnittstellen, die Sie zum Erstellen von Popup-Menüs verwenden können, sowie das Command Framework API, mit dem Sie den Code modularer aufbauen können. Wenn Ihre Anwendung zum Beispiel eine Benutzeroberflächenkomponente und ein Menüeintrag enthält, der die gleiche Funktion ausführt, können Sie diesen Funktions-Code mithilfe des Command Framework APIs einmal erfassen und in der gesamten Anwendung sowie in anderen Anwendungen zur Verfügung stellen.

Ein Kontextmenüeintrag enthielt eine Ordinalzahl, die im Menü die Platzierung des Elements festgelegt hat. Um so niedriger die Ordinalzahl, um so höher die Position des Elements im Menü. Wenn ein Kontextmenü in ein Popup-Menü umgewandelt wird, wird der Menüeintrag mit der niedrigsten Ordinalzahl zum standardmäßigen Popup-Menüeintrag. Die übrigen Elemente werden zum Popup-Menü von der niedrigsten Ordinalzahl zur höchsten hinzugefügt. Wenn ein Symbol mit einem Menüeintrag des älteren Kontextmenüs verknüpft war, wird es für das Element im Popup-Menü verwendet. Andernfalls wird ein Standardsymbol verwendet. Ähnlich zu anderen Popup-Menüs, werden nur die ersten acht Kontextmenüeinträge im Popup-Menü angezeigt. Wenn das Kontextmenü mehr als acht Menüeinträge beinhaltet, stellen Sie sicher, dass Sie die Ordinalzahlen für die Menüeinträge entsprechend verwenden, damit die wichtigsten und nützlichsten Elemente im Popup-Menü angezeigt werden.

Weitere Informationen zum Erstellen von Kontextmenüs finden Sie im *BlackBerry Developer Resource Center* unter www.blackberry.com/developer. Lesen Sie hier den Artikel [Distinguish between a full menu and a primary actions menu](#).

Erstellen eines Popup-Menüs

Sie können ein Popup-Menü mithilfe von Klassen erstellen, die im `net.rim.device.api.ui.menu`-Paket verfügbar sind. Außerdem können Sie die Funktionalität der Popup-Menüeinträge mithilfe des Command Framework APIs erstellen.

Ein Popup-Menü besteht aus den Lieferanten für Kontextmenü und Befehlselement und aus Befehlselementen.

Element	Beschreibung
Lieferant für das Kontextmenü	Sie verwenden die <code>DefaultContextMenuProvider</code> -Klasse, um das Popup-Menü eines Bildschirms zu erstellen und anzuzeigen. Wenn ein BlackBerry®-Gerätebenutzer ein Popup-Menü öffnet, sucht der Lieferant des Kontextmenüs nach Feldern auf dem Bildschirm, die Lieferanten für Befehlselemente sind. <code>DefaultContextMenuProvider</code> ist die Standardimplementierung von <code>ContextMenuProvider</code> . Wenn Sie keinen Bildschirm mit einem Lieferanten für ein Kontextmenü bereitstellen, wird das ältere Kontextmenü konvertiert und als Popup-Menü angezeigt.
Lieferant für Befehlselement	Sie verwenden die <code>CommandItemProvider</code> -Klasse, um ein Feld auf dem Benutzeroberflächen-Bildschirm zu konfigurieren, damit das Feld dem Popup-Menü Kontext und Befehlselemente basierend auf dem aktuellen Kontext bereitstellen kann. So könnten Sie beispielsweise eine E-Mail-Adresse als Lieferant eines Befehlselements konfigurieren und dem Benutzer verschiedene Aktionen bereitstellen, die darauf basieren, ob sich die E-Mail-Adresse in der Kontaktliste des Benutzers befindet oder nicht.
Befehlselemente	Sie verwenden die <code>CommandItem</code>-Klasse, um den Text sowie das Symbol und Verhalten eines Popup-Menüeintrags festzulegen. Sie definieren das Verhalten des Popup-Menüs mithilfe eines Befehls. Der Befehl fungiert als Proxy zu einer Befehlsverarbeitungs-Instanz, die die Funktionalität des Popup-Menüeintrags definiert. So könnten Sie beispielsweise Befehlselemente für eine E-Mail-Adresse bereitstellen, um einen Kontakt in Abhängigkeit davon, ob die ausgewählte E-Mail-Adresse in der Kontaktliste des Benutzers vorhanden ist, hinzuzufügen oder anzuzeigen. Die Befehlsverarbeitung würde den Code enthalten, um den Kontakt hinzuzufügen oder anzuzeigen. Weitere Informationen zu Befehlen und zur Befehlsverarbeitung finden Sie unter Command Framework API. In der im BlackBerry® Java® SDK bereitgestellten Command Framework-Beispielanwendung werden Befehle und Befehlsverarbeitungen veranschaulicht.

Wenn Sie ein älteres Kontextmenü verwenden, konvertiert BlackBerry® Device Software Device Software 6.0 die Kontextmenüeinträge in Befehlselemente und stellt diese dem Lieferant der Befehlselemente zur Verfügung. Weitere Informationen finden Sie unter [Unterstützung für ältere Kontextmenüs](#).

Erstellen eines Popup-Menüs

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.command.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
```

```
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.menu.*;
import net.rim.device.api.ui.image.*;
import net.rim.device.api.util.*;
import java.util.*;
```

- Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungskonstruktor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `MyPopUpMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class MyPopUpMenuApp extends UiApplication
{
    public static void main(String[] args)
    {
        Mypop-upMenuApp theApp = new Mypop-upMenuApp();
        theApp.enterEventDispatcher();
    }
    public Mypop-upMenuApp()
    {
        pushScreen(new Mypop-upMenuScreen());
    }
}
```

- Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben.

```
class MyPopUpMenuScreen extends MainScreen
{
    EmailAddressEditField emailAddress;
    public Mypop-upMenuScreen()
    {
        setTitle("Pop-Up Menu Demo");
    }
}
```

- Legen Sie im Bildschirmkonstruktor einen Lieferanten für das Kontextmenü fest. Übergeben Sie ein `DefaultContextMenuProvider`-Objekt an `Screen.setContextMenuProvider()`, um auf Ihrem Bildschirm die Anzeige eines Popup-Menüs zu ermöglichen.

```
setContextMenuProvider(new DefaultContextMenuProvider());
```

- Erstellen Sie im Bildschirmkonstruktor Benutzeroberflächen-Komponenten, die das Popup-Menü aufrufen können. Im folgenden Codebeispiel verwendet die Bezeichnung die `Field.FOCUSABLE`-Eigenschaft, um Benutzern zu erlauben, das Feld zu markieren.

```
LabelField labelField = new LabelField("Click to invoke pop-up menu",
    Field.FOCUSABLE);
emailAddress = new EmailAddressEditField("Email address: ", "name@blackberry.com",
    40);
```

6. Konfigurieren Sie im Bildschirmkonstruktor die Benutzeroberflächen-Komponenten als Lieferant der Befehlselemente. Das `DefaultContextMenuProvider`-Objekt sucht Felder, die als Lieferanten für Befehlselemente konfiguriert wurden und verwendet diese zur Erstellung und Anzeige eines Popup-Menüs. Rufen Sie für jede Komponente `Field.setCommandItemProvider()` auf, um das Feld als einen allgemeinen Befehlselementlieferanten zu konfigurieren. Die `ItemProvider`-Klasse wird in Schritt 7 beschrieben.

```
ItemProvider itemProvider = new ItemProvider();
labelField.setCommandItemProvider(itemProvider);
emailAddress.setCommandItemProvider(itemProvider);
```

7. Implementieren Sie auf dem benutzerdefinierten Bildschirm die `CommandItemProvider`-Schnittstelle. Geben Sie in `getContext()` das Feld zurück, das als Befehlselementlieferant konfiguriert wurde. Erstellen Sie in `getItems()` ein `Vector`-Objekt, dem die Popup-Menüeinträge hinzugefügt werden sollen.

```
class ItemProvider implements CommandItemProvider
{
    public Object getContext(Field field)
    {
        return field;
    }
    public Vector getItems(Field field)
    {
    }
}
```

8. Stellen Sie in `getItems()` die Popup-Menüeinträge bereit, indem Sie Instanzen der `CommandItem`-Klasse erstellen. Geben Sie für jedes Befehlselement den Popup-Menütext, das Symbol und den Befehl an. Im folgenden Codebeispiel wird eine `if-then-else`-Anweisung verwendet, um zu prüfen, welche Komponente das Popup-Menü vor der `CommandItem`-Erstellung aufgerufen hat. Der Befehl ist ein Proxy zu einer Klasseninstanz, die die abstrakte `CommandHandler`-Klasse erweitert, die in Schritt 9 beschrieben wird. Rufen Sie `Vector.addElement()` auf, um die Popup-Menüeinträge zum `Vector` hinzuzufügen. Geben Sie den `Vector` zurück.

```
CommandItem defaultCmd;
Image myIcon = ImageFactory.createImage(Bitmap.getBitmapResource("my_logo.png"));
if(field.equals(emailAddress)){
    defaultCmd = new CommandItem(new StringProvider("Email Address"), myIcon, new
        Command(new DialogCommandHandler()));
}
else {
    defaultCmd = new CommandItem(new StringProvider("Label Field"), myIcon, new
        Command(new DialogCommandHandler()));
}
items.addElement(defaultCmd);
return items;
```

9. Erstellen Sie auf dem benutzerdefinierten Bildschirm eine Befehlsverarbeitung, indem Sie eine Klasse erstellen, die die abstrakte `CommandHandler`-Klasse erweitert. Definieren Sie in `execute()` die Funktionalität, die Sie mit den Popup-Menüeinträgen verknüpfen möchten. Diese Befehlsverarbeitung könnte auf Ihrem Bildschirm von einer beliebigen Benutzeroberflächen-Komponente (zum Beispiel von Elementen in vollständigen Menüs, Schaltflächen usw.) verwendet werden. Da `canExecute()` nicht implementiert ist, ist dieser Befehl immer ausführbar. Im folgenden Codebeispiel wird ein Dialogfeld angezeigt, wenn der Benutzer auf einen Popup-Menüeintrag klickt.

```
class DialogCommandHandler extends CommandHandler
{
    public void execute(ReadOnlyCommandMetadata metadata, Object context)
    {
        Dialog.alert("Executing command for " + context.toString());
    }
}
```

Codebeispiel: Erstellen eines Popup-Menüs

```
import net.rim.device.api.command.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.menu.*;
import net.rim.device.api.ui.image.*;
import net.rim.device.api.util.*;
import java.util.*;
public class MyPopupMenuApp extends UiApplication
{
    public static void main(String[] args)
    {
        MyPopupMenuApp theApp = new MyPopupMenuApp();
        theApp.enterEventDispatcher();
    }
    public MyPopupMenuApp()
    {
        pushScreen(new MyPopupMenuScreen());
    }
}
class MyPopupMenuScreen extends MainScreen
{
    EmailAddressEditField emailAddress;
    public MyPopupMenuScreen()
    {
        setTitle("Pop-Up Menu Demo");
        setContextMenuProvider(new DefaultContextMenuProvider());
        LabelField labelField = new LabelField("Click to invoke pop-up menu",
        Field.FOCUSABLE);
        emailAddress = new EmailAddressEditField("Email address: ",
        "name@blackberry.com", 40);
        ItemProvider itemProvider = new ItemProvider();
        labelField.setCommandItemProvider(itemProvider);
        emailAddress.setCommandItemProvider(itemProvider);
        add(labelField);
        add(emailAddress);
    }
    /* To override the default functionality that prompts the user to save changes
    before the application closes,
    * override the MainScreen.onSavePrompt() method. In the following code sample,
    the return value is true which
    * indicates that the application does not prompt the user before closing.
    */
    protected boolean onSavePrompt()
```

```

{
    return true;
}
class ItemProvider implements CommandItemProvider
{
    public Object getContext(Field field)
    {
        return field;
    }
    public Vector.getItems(Field field)
    {
        Vector items = new Vector();
        CommandItem defaultCmd;
        Image myIcon = ImageFactory.createImage(Bitmap
.getBitmapResource("my_logo.png"));
        if(field.equals(emailAddress)){
            defaultCmd = new CommandItem(new StringProvider("Email Address"),
myIcon, new Command(new DialogCommandHandler()));
        }
        else{
            defaultCmd = new CommandItem(new StringProvider("Label Field"),
myIcon, new Command(new DialogCommandHandler()));
        }
        items.addElement(defaultCmd);
        return items;
    }
}
class DialogCommandHandler extends CommandHandler
{
    public void execute(ReadOnlyCommandMetadata metadata, Object context)
    {
        Dialog.alert("Executing command for " + context.toString());
    }
}
}

```

Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software

Sie können einer BlackBerry® Device Software-Anwendung einen Menüeintrag mithilfe des APIs für Menüeinträge im `net.rim.blackberry.api.menuitem`-Paket hinzufügen. Zum Beispiel können Sie einen Menüeintrag namens "View Sales Order" zur Kontaktanwendung auf einem BlackBerry hinzufügen, sodass eine CRM-Anwendung mit einer Liste von Vertriebsaufträgen für den Kontakt geöffnet und angezeigt wird, sobald ein Benutzer auf den Menüeintrag klickt.

Die `ApplicationMenuItemRepository`-Klasse stellt die Konstanten bereit, die die BlackBerry Device Software-Anwendung festlegen, in der Ihr Menüeintrag angezeigt werden soll. Die `MENUITEM_MESSAGE_LIST`-Konstante gibt zum Beispiel an, dass der Menüeintrag in der Nachrichtenanwendung angezeigt werden soll.

Hinzufügen eines Menüeintrags zu einer BlackBerry Device Software-Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.blackberry.api.menuitem.*;
import net.rim.blackberry.api.pdap.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
```

2. Erweitern Sie die abstrakte `ApplicationMenuItem`-Klasse, um einen Menüeintrag zu erstellen. Überschreiben Sie den `ApplicationMenuItem()`-Konstruktor mit einer Ganzzahl, um die Position des Menüeintrags im Menü anzugeben. Durch eine höhere Zahl wird der Menüeintrag weiter unten im Menü platziert.

```
public class SampleMenuItem extends ApplicationMenuItem
{
    SampleMenuItem()
    {
        super(20);
    }
}
```

3. Implementieren Sie `toString()`, um den Text des Menüeintrags festzulegen.

```
public String toString()
{
    return "Open the Contacts Demo application";
}
```

4. Rufen Sie `getInstance()` auf, um die Anwendungsdatenbank abzurufen.

```
ApplicationMenuItemRepository repository =
    ApplicationMenuItemRepository.getInstance();
```

5. Erstellen Sie eine Instanz einer Klasse, um die `MenuItem`-Klasse zu erweitern.

```
ContactsDemoMenuItem contactsDemoMenuItem = new
    ContactsDemoMenuItem();
```

6. Rufen Sie `ApplicationMenuItemRepository.addItem()` auf, um den Menüeintrag zur entsprechenden Anwendungsdatenbank auf dem BlackBerry®-Gerät hinzuzufügen.

```
repository.addItem(ApplicationMenuItemRepository
    .MENUITEM_ADDRESSCARD_VIEW, contactsDemoMenuItem);
```

7. Implementieren Sie `run()`, um das Verhalten des Menüeintrags festzulegen. Im folgenden Codebeispiel ruft die `ContactsDemo`-Anwendung das Ereignis und `ContactsDemo.enterEventDispatcher()` auf, wenn ein Benutzer auf den neuen Menüeintrag klickt und ein `Contact`-Objekt vorhanden ist.

```
public Object run(Object context)
{
    BlackBerryContact c = (BlackBerryContact)context;
    if ( c != null )
    {
        new ContactsDemo().enterEventDispatcher();
    }
}
```

```

    }
    else
    {
        throw new IllegalStateException( "Context is null, expected a Contact
        instance");
    }
    Dialog.alert("Viewing an email message in the email view");
    return null;
}

```

Ändern der Darstellung eines Menüs

Sie können den Hintergrund, Rahmen und die Schriftart eines Menüs mithilfe der Menu-Klasse im `net.rim.device.api.ui.component`-Paket ändern. Zum Beispiel können Sie die Anzeige des Menüs ändern, damit es ein Erscheinungsbild hat, das dem Rest Ihrer BlackBerry®-Geräteanwendung ähnlich ist. Wenn Sie die Anzeige eines Menüs ändern, hebt Ihre BlackBerry-Geräteanwendung das Thema auf, das auf dem BlackBerry-Gerät festgelegt ist.

Ändern der Darstellung eines Menüs

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.system.*;

```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie in `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um der Anwendung das Empfangen von Ereignissen zu ermöglichen. Rufen Sie im Anwendungskonstruktor `pushScreen()` auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `CreateCustomMenuScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```

public class CreateCustomMenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateCustomMenu theApp = new CreateCustomMenu();
        theApp.enterEventDispatcher();
    }
    public CreateCustomMenu()
    {
        pushScreen(new CreateCustomMenuScreen());
    }
}

```

3. Erstellen Sie den benutzerdefinierten Bildschirm für die Anwendung durch Erweitern der `MainScreen`-Klasse. Rufen Sie im Bildschirmkonstruktor `setTitle()` auf, um den Titel für den Bildschirm anzugeben. Rufen Sie `add()` auf, um ein Textfeld auf dem Bildschirm hinzuzufügen.

```
class CreateCustomMenuScreen extends MainScreen
{
    Background _menuBackground;
    Border _menuBorder;
    Font _menuFont;
    CreateCustomMenuScreen()
    {
        setTitle("Custom Menu Sample");
        add(new RichTextField("Creating a custom menu"));
    }
}
```

4. Geben Sie im Bildschirmkonstruktor die Darstellung des Menüs an. Erstellen Sie den Abstand für den Menürahmen durch Erstellen eines XYEdges-Objekts. Rufen Sie createRoundedBorder() auf, um einen Rahmen mit abgerundeten Ecken zu erstellen. Rufen Sie createSolidTransparentBackground() auf, um eine transparente Hintergrundfarbe für das Menü zu erstellen.

```
XYEdges thickPadding = new XYEdges(10, 10, 10, 10);
_menuBorder = BorderFactory.createRoundedBorder(thickPadding,
Border.STYLE_DOTTED);
_menuBackground = BackgroundFactory.createSolidTransparentBackground(Color
.LIGHTSTEELBLUE, 50);
```

5. Legen Sie im Bildschirmkonstruktor die Schriftart für das Menü mithilfe eines FontFamily-Objekts fest. Rufen Sie forName() auf, um eine Schriftart vom BlackBerry®-Gerät abzurufen. Rufen Sie getFont() auf, um den Stil und die Größe der Schriftart anzugeben.

```
try
{
    FontFamily family = FontFamily.forName("BBCasual");
    _menuFont = family.getFont(Font.PLAIN, 30, Ui.UNITS_px);
}
catch(final ClassNotFoundException cnfe)
{
    UiApplication.getUiApplication().invokeLater(new Runnable()
    {
        public void run()
        {
            Dialog.alert("FontFamily.forName() threw " + cnfe.toString());
        }
    });
}
```

6. Überschreiben Sie in der Screen-Klasse makeMenu(), um die Darstellung des Menüs anzuwenden. Rufen Sie setBackground(), setBorder() und setFont() auf, um die Darstellung des Menüs anzuwenden, die Sie in den Schritten 4 und 5 angegeben haben.

```
protected void makeMenu(Menu menu, int context)
{
    menu.setBorder(_menuBorder);
    menu.setBackground(_menuBackground);
    menu.setFont(_menuFont);
    super.makeMenu(menu, context);
}
```

Codebeispiel: Ändern der Darstellung eines Menüs

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.decor.*;
import net.rim.device.api.system.*;
public class CreateCustomMenu extends UiApplication
{
    public static void main(String[] args)
    {
        CreateCustomMenu theApp = new CreateCustomMenu();
        theApp.enterEventDispatcher();
    }
    public CreateCustomMenu()
    {
        pushScreen(new CreateCustomMenuScreen());
    }
}
class CreateCustomMenuScreen extends MainScreen
{
    Border _menuBorder;
    Background _menuBackground;
    Font _menuFont;
    CreateCustomMenuScreen()
    {
        setTitle("Custom Menu Sample");
        add(new RichTextField("Creating a custom menu"));
        XYEdges thickPadding = new XYEdges(10, 10, 10, 10);
        _menuBorder = BorderFactory.createRoundedBorder(thickPadding,
        Border.STYLE_DOTTED);
        _menuBackground = BackgroundFactory.createSolidTransparentBackground(Color
        .LIGHTSTEELBLUE, 50);
        try
        {
            FontFamily family = FontFamily.forName("BBCasual");
            _menuFont = family.getFont(Font.PLAIN, 30, Ui.UNITS_px);
        }
        catch(final ClassNotFoundException cnfe)
        {
            UiApplication.getUiApplication().invokeLater(new Runnable()
            {
                public void run()
                {
                    Dialog.alert("FontFamily.forName() threw " + cnfe.toString());
                }
            });
        }
    }
    protected void makeMenu(Menu menu, int context)
    {
        menu.setBorder(_menuBorder);
        menu.setBackground(_menuBackground);
        menu.setFont(_menuFont);
    }
}
```

```
        super.makeMenu(menu, context);  
    }  
}
```

Benutzerdefinierte Schriftarten

11

Die `FontManager`-Klasse im `net.rim.device.api.ui`-Paket stellt Konstanten und Methoden bereit, die Sie verwenden können, um eine TrueType-Schriftart auf einem BlackBerry®-Gerät zu installieren und zu deinstallieren. Die maximale Größe, die für die TrueType-Schriftartdatei ermöglicht wird, ist 60 KB. Sie können angeben, ob die Schriftart für die Anwendung, die die Schriftart installiert oder für alle Anwendungen auf dem BlackBerry-Gerät verfügbar ist.

Die `FontManager`-Klasse stellt auch Methoden bereit, um die Standardschriftart für das BlackBerry-Gerät oder die Anwendung festzulegen.

Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry-Java-Anwendung

1. Importieren Sie die erforderlichen Klassen und Schnittstellen.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
import java.util.*;
```

2. Erstellen Sie das Anwendungsframework durch Erweitern der `UiApplication`-Klasse. Erstellen Sie unter `main()` eine Instanz der neuen Klasse, und rufen Sie `enterEventDispatcher()` auf, um die Anwendung für das Empfangen von Ereignissen zu aktivieren. Rufen Sie `pushScreen()` im Anwendungsconstructor auf, um den benutzerdefinierten Bildschirm für die Anwendung anzuzeigen. Die in Schritt 3 beschriebene `FontLoadingDemoScreen`-Klasse stellt den benutzerdefinierten Bildschirm dar.

```
public class FontLoadingDemo extends UiApplication
{
    public static void main(String[] args)
    {
        FontLoadingDemo app = new FontLoadingDemo();
        app.enterEventDispatcher();
    }
    public FontLoadingDemo()
    {
        pushScreen(new FontLoadingDemoScreen());
    }
}
```

3. Erstellen Sie den benutzerdefinierten Bildschirm durch Erweitern der `MainScreen`-Klasse. Rufen Sie `setTitle()` auf, um den Text festzulegen, der im Titelbereich des Bildschirms angezeigt wird. Erstellen Sie ein neues `LabelField`-Objekt. Sie wenden die benutzerdefinierte Schriftart auf dieses Objekt an.

```
class FontLoadingDemoScreen extends MainScreen
{
    public FontLoadingDemoScreen()
    {
        setTitle("Font Loading Demo");
    }
}
```

```

        LabelField helloWorld = new LabelField("Hello World");
    }
}

```

4. Rufen Sie im Bildschirmkonstruktor die `FontManager.getInstance()`-Methode auf, um einen Verweis auf das `FontManager`-Objekt zu erhalten, und rufen Sie anschließend die `load()`-Methode auf, um die Schriftart zu installieren. Schließen Sie den `load()`-Aufruf in eine IF-Anweisung ein, um zu prüfen, ob die Installation erfolgreich war. Die `load()`-Methode gibt eine Markierung zurück, die angibt, ob die Schriftart erfolgreich installiert wurde. Der folgende Code gibt an, dass die Schriftart nur von der Anwendung verwendet werden kann.

```

if (FontManager.getInstance().load("Myfont.ttf", "MyFont",
    FontManager.APPLICATION_FONT) == FontManager.SUCCESS)
{
}

```

5. Erstellen Sie im Bildschirmkonstruktor in einem try/catch-Block in der in Schritt 5 erstellten IF-Anweisung ein `Font`-Objekt für die Schriftart, die Sie gerade installiert haben. Rufen Sie die `setFont()`-Methode auf, um die Schriftart auf das in Schritt 5 erstellte `LabelField` anzuwenden.

```

try
{
    FontFamily family = FontFamily.forName("MyFont");
    Font myFont = family.getFont(Font.PLAIN, 50);
    helloWorld.setFont(myFont);
}
catch (ClassNotFoundException e)
{
    System.out.println(e.getMessage());
}

```

6. Rufen Sie im Bildschirmkonstruktor `add()` auf, um dem Bildschirm `LabelField` hinzuzufügen.

```

add(helloWorld);

```

Codebeispiel: Installieren und Verwenden einer benutzerdefinierten Schriftart in einer BlackBerry Java-Anwendung

```

import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.component.*;
import java.util.*;
public class FontLoadingDemo extends UiApplication
{
    public static void main(String[] args)
    {
        FontLoadingDemo app = new FontLoadingDemo();
        app.enterEventDispatcher();
    }
    public FontLoadingDemo()
    {

```

```
        pushScreen(new FontLoadingDemoScreen());
    }
}
class FontLoadingDemoScreen extends MainScreen
{
    public FontLoadingDemoScreen()
    {
        setTitle("Font Loading Demo");
        LabelField helloWorld = new LabelField("Hello World");
        if (FontManager.getInstance().load("Myfont.ttf", "MyFont",
            FontManager.APPLICATION_FONT) == FontManager.SUCCESS)
        {
            try
            {
                FontFamily typeface = FontFamily.forName("MyFont");
                Font myFont = typeface.getFont(Font.PLAIN, 50);
                helloWorld.setFont(myFont);
            }
            catch (ClassNotFoundException e)
            {
                System.out.println(e.getMessage());
            }
        }
        add(helloWorld);
    }
}
```

Rechtschreibprüfung

12

Sie können die Elemente im `net.rim.blackberry.api.spellcheck`-Paket verwenden, um einer Anwendung Rechtschreibprüfungsfunktionen hinzuzufügen. Die `SpellCheckEngine`-Schnittstelle befähigt eine Anwendung, die Rechtschreibung eines UI-Feld-Wertes zu prüfen, und gibt einem BlackBerry®-Gerätebenutzer die Optionen zur Rechtschreibkorrektur. Die `SpellCheckUI`-Schnittstelle befähigt eine Anwendung, eine Benutzeroberfläche bereitzustellen, die es einem BlackBerry-Gerätebenutzer ermöglicht, ein Rechtschreibproblem durch Interagieren mit der `SpellCheckEngine`-Implementierung zu lösen.

Weitere Informationen zur Verwendung der Spell-Check-API können Sie der Spell-Check-Beispielanwendung entnehmen, die in der BlackBerry® Java® Development Environment 4.3.1 oder neuer sowie im BlackBerry® Java® Plug-in für Eclipse® enthalten ist.

Hinzufügen einer Funktion zur Rechtschreibprüfung

1. Importieren Sie die folgenden Klassen:
 - `net.rim.blackberry.api.spellcheck.SpellCheckEngineFactory`
 - `java.lang.StringBuffer`
2. Importieren Sie die folgenden Schnittstellen:
 - `net.rim.blackberry.api.spellcheck.SpellCheckEngine`
 - `net.rim.blackberry.api.spellcheck.SpellCheckUI`
 - `net.rim.blackberry.api.spellcheck.SpellCheckUIListener`
3. Erstellen Sie Variablen für Rechtschreibprüfungsobjekte.

```
SpellCheckEngine _spellCheckEngine;  
SpellCheckUI _spellCheckUI;
```

4. Rufen Sie `createSpellCheckUI()` auf.

```
_spellCheckUI = SpellCheckEngineFactory.createSpellCheckUI();
```

5. Um eine Anwendung zu benachrichtigen, wenn ein Rechtschreibprüfungsereignis auftritt, rufen Sie `addSpellCheckUIListener()` mit einem `SpellCheckUIListener`-Objekt als Parameter auf.

```
_spellCheckUI.addSpellCheckUIListener(new SpellCheckUIListener());
```

6. Um eine Anwendung eine Rechtschreibprüfung für Benutzeroberflächenfelder durchführen und BlackBerry-Gerätebenutzern Rechtschreibkorrekturen vorschlagen zu lassen, rufen Sie ein `SpellCheckEngine`-Objekt ab, und rufen Sie `getSpellCheckEngine()` auf.

```
_spellCheckEngine = _spellCheckUI.getSpellCheckEngine();
```

7. Um eine Korrektur für ein falsch buchstabiertes Wort zu verwenden, rufen Sie `SpellCheckEngine.learnCorrection()` auf. Verwenden Sie die Parameter `new StringBuffer(text)`, `new StringBuffer(correction)`, wobei `text` das falsch buchstabierte Wort und `correction` das richtige Wort darstellt.

```
_spellCheckEngine.learnCorrection(new StringBuffer(text), new StringBuffer(correction));
```

8. Um für ein Feld Rechtschreibprüfungsvorgänge auszuführen, rufen Sie `SpellCheckUI.spellCheck()` mit `field` als Parameter auf.

```
_spellCheckUI.spellCheck(field);
```

9. Um ein falsch buchstabiertes Wort als richtig buchstabiertes Wort zu akzeptieren, rufen Sie `SpellCheckEngine.learnWord()` mit dem zu lernenden Wort als Parameter auf.

```
_spellCheckEngine.learnWord(new StringBuffer(word));
```

Empfangen von Rechtschreibprüfungsereignissen

1. Importieren Sie die folgenden Klassen:
 - `java.lang.StringBuffer`
 - `net.rim.device.api.ui.UiApplication`
 - `net.rim.device.api.ui.Field`
2. Importieren Sie die folgenden Schnittstellen:
 - `net.rim.blackberry.api.spellcheck.SpellCheckUIListener`
 - `net.rim.blackberry.api.spellcheck.SpellCheckEngine`
3. Erstellen Sie eine Methode, die die `SpellCheckUIListener.LEARNING_ACCEPT`-Konstante zurückgibt, wenn `SpellCheckEngine` ein neues Wort lernt.

```
public int wordLearned(SpellCheckUI ui, StringBuffer word) {
    UiApplication.getUiApplication().invokeLater(new popUpRunner("Word learned"));
    return SpellCheckUIListener.LEARNING_ACCEPT;
}
```

4. Erstellen Sie eine Methode, die die `SpellCheckUIListener.LEARNING_ACCEPT`-Konstante zurückgibt, wenn `SpellCheckEngine` eine Korrektur für ein Wort lernt.

```
public int wordCorrectionLearned(SpellCheckUI ui, StringBuffer word, StringBuffer correction){
    UiApplication.getUiApplication().invokeLater(new popUpRunner("Correction learned"));
    return SpellCheckUIListener.LEARNING_ACCEPT;
}
```

5. Erstellen Sie eine Methode, die die `SpellCheckUIListener.ACTION_OPEN_UI`-Konstante zurückgibt, wenn `SpellCheckEngine` ein falsch buchstabiertes Wort findet.

```
public int misspelledWordFound(SpellCheckUI ui, Field field, int offset, int len){
    UiApplication.getUiApplication().invokeLater(new popUpRunner("Misspelled word found"));
    return SpellCheckUIListener.ACTION_OPEN_UI;
}
```

Verwandte Ressourcen

13

- www.blackberry.com/go/apiref: Zeigen Sie die aktuelle Version der API-Referenz für das BlackBerry® Java® SDK an.
- www.blackberry.com/go/devguides: Hier finden Sie Entwicklungshandbücher, Versionshinweise und Überblicke über die Beispielanwendung für das BlackBerry Java SDK.
- www.blackberry.com/developers: Besuchen Sie die BlackBerry® Developer Zone für Ressourcen zur Entwicklung von BlackBerry-Geräteanwendungen.
- www.blackberry.com/go/developerkb: Zeigen Sie Knowledge Base-Artikel der BlackBerry Development Knowledge Base an.
- www.blackberry.com/developers/downloads: Hier finden Sie die aktuellen Entwicklungstools und Downloads zum Entwickeln von BlackBerry-Geräteanwendungen.

Glossar

14

3D

Dreidimensional

API

Application Programming Interface (Anwendungsprogrammierschnittstelle)

JVM

Java® Virtual Machine

MIDP

Mobile Information Device Profile (Softwareprofil für mobile Geräte)

Rückmeldung

15

Falls Sie uns eine Rückmeldung zu diesem Artikel zukommen lassen möchten, besuchen Sie die Website www.blackberry.com/docsfeedback.

Verlauf der Dokumentrevision

16

Datum	Beschreibung
16. August 2010	Das folgende Thema wurde entfernt: • Erstellen einer Fortschrittsanzeige
3. August 2010	Die folgenden Themen wurden hinzugefügt: • Zugriff auf ein codiertes Bild über einen Eingabestream • Info über das Einfügen von Elementen in Popup-Menüs • Bewährtes Verfahren: Implementieren von Popup-Menüs • Bewährtes Verfahren: Implementieren von Untermenüs • Command Framework API • Erstellen eines Popup-Menüs • Erstellen eines Popup-Menüs • Erstellen eines Untermenüs • Anzeigen einer Bezeichnung an einer absoluten Position auf dem Bildschirm • Anzeigen eines codierten Bilds • Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken • Anzeigen mehrerer Bilder zum Scrollen • Anzeigen eines Felds an einer absoluten Position auf dem Bildschirm • Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken • Anzeigen mehrerer Bilder zum Scrollen • Aktivieren von Pinch-Gesten • Aktivieren von Fingerstrich-Gesten, die Benutzer auf dem Trackpad machen • Codieren eines Bilds • Bilder • Anzeigen der Aktivität • Anzeigen des Fortschritts • Anzeigen von Aktivität oder Aufgabenfortschritt • Pinch-Gesten • Popup-Menüs • Festlegen des Decodierungsmodus für ein Bild • Festlegen der Anzeigegröße eines codierten Bilds • Untermenüs • Unterstützung für ältere Kontextmenüs • Fingerstrich-Gesten mit Trackpads • Modelle für Touchscreen-Interaktionen

Datum	Beschreibung
	• Verwenden eines Befehl in mindestens einer Anwendung • Verwenden eines Befehls mit einer Benutzeroberflächenkomponente • Verwenden von codierten Bildern Die folgenden Codebeispiele wurden hinzugefügt: • Codebeispiel: Erstellen eines Popup-Menüs • Codebeispiel: Erstellen eines Untermenüs • Codebeispiel: Anzeigen einer Reihe von Bildern zum Scrollen • Codebeispiel: Anzeigen einer Bezeichnung an einer absoluten Bildschirmposition • Codebeispiel: Anzeigen eines Bilds zum Vergrößern/Verkleinern und Schwenken Die folgenden Themen wurden geändert: • Aktivitäts- und Fortschrittsanzeigen • Schaltflächen • Erstellen einer Benutzeroberfläche, die mit den standardmäßigen BlackBerry-Benutzeroberflächen verträglich ist • Dialogfelder • Dropdown-Listen • Listen und Tabellen • Auswahltool • Optionsfelder • Suche • Drehfelder • Textfelder • Strukturansichten Die folgenden Themen wurden entfernt: • Hinzufügen eines Symbols zu einem Menüeintrag • Hinzufügen eines Symbols zu einem Menüeintrag • Codebeispiel: Hinzufügen eines Symbols zu einem Menüeintrag

Rechtliche Hinweise

17

©2010 Research In Motion Limited. Alle Rechte vorbehalten. BlackBerry®, RIM®, Research In Motion® und die zugehörigen Marken, Namen und Logos sind Eigentum von Research In Motion Limited und sind in den USA und weiteren Ländern weltweit als Marken eingetragen und/oder werden dort als Marken verwendet.

Eclipse ist eine Marke der Eclipse Foundation, Inc. Java ist eine Marke von Oracle America, Inc. TrueType ist eine Marke von Apple Inc. Alle weiteren Marken sind Eigentum ihrer jeweiligen Inhaber.

Dieses Dokument und alle Dokumente, die per Verweis in dieses Dokument mit einbezogen werden, z. B. alle unter www.blackberry.com/go/docs erhältlichen Dokumente, werden ohne Mängelgewähr und je nach Verfügbarkeit bereitgestellt. Die entsprechenden Dokumente werden ohne ausdrückliche Billigung, Gewährleistung oder Garantie seitens Research In Motion Limited und seinen angegliederten Unternehmen („RIM“) bereitgestellt. RIM übernimmt keine Verantwortung für eventuelle typografische, technische oder anderweitige Ungenauigkeiten sowie für Fehler und Auslassungen in den genannten Dokumenten. Dieses Dokument beschreibt eventuell einige Aspekte der RIM-Technologie in allgemeiner Form, um das Eigentum und die vertraulichen Informationen und/oder Geschäftsgeheimnisse von RIM zu schützen. RIM behält sich das Recht vor, die in diesem Dokument enthaltenen Informationen von Zeit zu Zeit zu ändern. RIM ist jedoch nicht verpflichtet, die Benutzer über diese Änderungen, Updates, Verbesserungen oder Zusätze rechtzeitig bzw. überhaupt in Kenntnis zu setzen.

Diese Dokumentation enthält möglicherweise Verweise auf Informationsquellen, Hardware oder Software, Produkte oder Dienste, einschließlich Komponenten und Inhalte wie urheberrechtlich geschützte Inhalte und/oder Websites von Drittanbietern (nachfolgend "Drittprodukte und -dienste" genannt). RIM hat keinen Einfluss auf und übernimmt keine Haftung für Drittprodukte und -dienste, dies gilt u. a. für Inhalt, Genauigkeit, Einhaltung der Urheberrechtsgesetze, Kompatibilität, Leistung, Zuverlässigkeit, Rechtmäßigkeit, Schicklichkeit, Links oder andere Aspekte der Drittprodukte und -dienste. Der Einschluss eines Verweises auf Drittprodukte und -dienste in dieser Dokumentation impliziert in keiner Weise eine besondere Empfehlung der Drittprodukte und -dienste oder des Drittanbieters durch RIM.

SO FERN ES NICHT DURCH DAS IN IHREM RECHTSGEBIET GELTENDE RECHT AUSDRÜCKLICH UNTERSAGT IST, WERDEN HIERMIT SÄMTLICHE AUSDRÜCKLICHEN ODER KONKLUDENTEN BEDINGUNGEN, BILLIGUNGEN, GARANTIEN, ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN JEDER ART, EINSCHLIESSLICH, OHNE EINSCHRÄNKUNG, BEDINGUNGEN, BILLIGUNGEN, GARANTIEN, ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN HINSICHTLICH DER HALTBARKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ODER VERWENDUNGSZWECK, MARKTGÄNGIGKEIT, MARKTGÄNGIGEN QUALITÄT, NICHTVERLETZUNG VON RECHTEN DRITTER, ZUFRIEDENSTELLENDEN QUALITÄT ODER DES EIGENTUMSRECHTS ABGELEHNT. DIES GILT AUCH FÜR ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN, DIE SICH AUS EINEM GESETZ, EINER GEPFLOGENHEIT, USANCEN BZW. HANDELSGEPFLOGENHEITEN ERGEBEN ODER IM ZUSAMMENHANG MIT DER DOKUMENTATION ODER IHRER VERWENDUNG, DER LEISTUNG ODER MANGELNDEN LEISTUNG VON SOFTWARE, HARDWARE, DIENSTEN ODER DRITTANBIETER-PRODUKTEN UND -DIENSTEN STEHEN, AUF DIE HIER VERWIESEN WIRD. MÖGLICHERWEISE HABEN SIE ZUDEM ANDERE LANDESSPEZIFISCHE RECHTE. IN MANCHEN RECHTSGEBIETEN IST DER AUSSCHLUSS ODER DIE EINSCHRÄNKUNG KONKLUDENTER GEWÄHRLEISTUNGEN UND BEDINGUNGEN NICHT ZULÄSSIG. IN DEM GESETZLICH ZULÄSSIGEN UMFANG WERDEN SÄMTLICHE KONKLUDENTEN GEWÄHRLEISTUNGEN ODER BEDINGUNGEN IM ZUSAMMENHANG MIT DER

DOKUMENTATION, DIE EINGESCHRÄNKT WERDEN KÖNNEN, SOFERN SIE NICHT WIE OBEN DARGELEGT AUSGESCHLOSSEN WERDEN KÖNNEN, HIERMIT AUF 90 TAGE AB DATUM DES ERWERBS DER DOKUMENTATION ODER DES ARTIKELS, AUF DEN SICH DIE FORDERUNG BEZIEHT, BESCHRÄNKT.

IN DEM DURCH DAS IN IHREM RECHTSGEBIET ANWENDBARE GESETZ MAXIMAL ZULÄSSIGEN AUSMASS HAFTET RIM UNTER KEINEN UMSTÄNDEN FÜR SCHÄDEN JEDLICHER ART, DIE IM ZUSAMMENHANG MIT DIESER DOKUMENTATION ODER IHRER VERWENDUNG, DER LEISTUNG ODER NICHTLEISTUNG JEDLICHER SOFTWARE, HARDWARE, DIENSTE ODER DRITTPRODUKTE UND -DIENSTE, AUF DIE HIER BEZUG GENOMMEN WIRD, STEHEN, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF DIE FOLGENDEN SCHÄDEN: DIREKTE, VERSCHÄRFTEN SCHADENERSATZ NACH SICH ZIEHENDE, BEILÄUFIG ENTSTANDENE, INDIREKTE, KONKRETE, STRAFE EINSCHLIESSENDE SCHÄDEN, FOLGESCHÄDEN ODER SCHÄDEN, FÜR DIE ANSPRUCH AUF KOMPENSATORISCHEN SCHADENERSATZ BESTEHT, SCHÄDEN WEGEN ENTGANGENEN GEWINNEN ODER EINKOMMEN, NICHTREALISIERUNG ERWARTETER EINSPARUNGEN, BETRIEBSUNTERBRECHUNGEN, VERLUSTES GESCHÄFTLICHER DATEN, ENTGANGENER GESCHÄFTSCHANCEN ODER BESCHÄDIGUNG BZW. VERLUSTES VON DATEN, DES UNVERMÖGENS, DATEN ZU ÜBERTRAGEN ODER ZU EMPFANGEN, PROBLEMEN IM ZUSAMMENHANG MIT ANWENDUNGEN, DIE IN VERBINDUNG MIT RIM-PRODUKTEN UND -DIENSTEN VERWENDET WERDEN, KOSTEN VON AUSFALLZEITEN, NICHTVERWENDBARKEIT VON RIM-PRODUKTEN UND -DIENSTEN ODER TEILEN DAVON BZW. VON AIRTIME-DIENSTEN, KOSTEN VON ERSATZGÜTERN, DECKUNG, EINRICHTUNGEN ODER DIENSTEN, KAPITAL- ODER ANDERE VERMÖGENSSCHÄDEN, UNABHÄNGIG DAVON, OB SCHÄDEN DIESER ART ABZUSEHEN ODER NICHT ABZUSEHEN WAREN, UND AUCH DANN, WENN RIM AUF DIE MÖGLICHKEIT SOLCHER SCHÄDEN HINGEWIESEN WURDE.

IN DEM DURCH DAS IN IHREM RECHTSGEBIET ANWENDBARE GESETZ MAXIMAL ZULÄSSIGEN AUSMASS ÜBERNIMMT RIM KEINERLEI VERANTWORTUNG, VERPFLICHTUNG ODER HAFTUNG, SEI SIE VERTRAGLICHER, DELIKTRECHTLICHER ODER ANDERWEITIGER NATUR, EINSCHLIESSLICH DER HAFTUNG FÜR FAHRLÄSSIGKEIT UND DER DELIKTSHAFTUNG.

DIE IN DIESEM DOKUMENT GENANNTE EINSCHRÄNKUNGEN, AUSSCHLÜSSE UND HAFTUNGSAUSSCHLÜSSE GELTEN (A) UNGEACHTET DER VON IHNEN ANGEFÜHRTEN KLAGEGRÜNDE, FORDERUNGEN ODER KLAGEN, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF VERTRAGSBRUCH, FAHRLÄSSIGKEIT, ZIVILRECHTLICHER DELIKTE, DELIKTSHAFTUNG ODER SONSTIGE RECHTSTHEORIE UND SIND AUCH NACH EINEM WESENTLICHEN VERSTOSS BZW. EINEM FEHLENDEN GRUNDLEGENDEN ZWECK DIESER VEREINBARUNG ODER EINES DARIN ENTHALTENEN RECHTSBEHELFS WIRKSAM; UND GELTEN (B) FÜR RIM UND DIE ZUGEHÖRIGEN UNTERNEHMEN, RECHTSNACHFOLGER, BEVOLLMÄCHTIGTEN, VERTRETER, LIEFERANTEN (EINSCHLIESSLICH AIRTIME-DIENSTANBIETERN), AUTORISIERTE RIM-DISTRIBUTOREN (EBENFALLS EINSCHLIESSLICH AIRTIME-DIENSTANBIETERN) UND DIE JEWEILIGEN FÜHRUNGSKRÄFTE, ANGESTELLTEN UND UNABHÄNGIGEN AUFTRAGNEHMER.

ZUSÄTZLICH ZU DEN OBEN GENANNTE EINSCHRÄNKUNGEN UND AUSSCHLÜSSEN HAFTEN DIE FÜHRUNGSKRÄFTE, ANGESTELLTEN, VERTRETER, DISTRIBUTOREN, LIEFERANTEN, UNABHÄNGIGEN AUFTRAGNEHMER VON RIM ODER RIM ANGEHÖRENDE UNTERNEHMEN IN KEINER WEISE IM ZUSAMMENHANG MIT DER DOKUMENTATION.

Bevor Sie Drittprodukte bzw. -dienste abonnieren, installieren oder verwenden, müssen Sie sicherstellen, dass Ihr Airtime-Dienstanbieter sich mit der Unterstützung aller zugehörigen Funktionen einverstanden erklärt hat. Einige Airtime-Dienstanbieter bieten möglicherweise keine Internet-Browsing-Funktion in Zusammenhang mit einem Abonnement für den BlackBerry® Internet Service an. Erkundigen Sie sich bei Ihrem Service Provider bezüglich Verfügbarkeit, Roaming-Vereinbarungen, Service-Plänen und Funktionen. Für die Installation und Verwendung von Drittprodukten und -diensten mit den Produkten und Diensten von RIM sind möglicherweise Patent-, Marken-,

Urheberrechts- oder andere Lizenzen erforderlich, damit die Rechte Dritter nicht verletzt werden. Es liegt in Ihrer Verantwortung, zu entscheiden, ob Sie Drittprodukte und -dienste verwenden möchten, und festzustellen, ob hierfür Lizenzen erforderlich sind. Für den Erwerb etwaiger Lizenzen sind Sie verantwortlich. Installieren oder verwenden Sie Drittprodukte und -dienste erst nach dem Erwerb aller erforderlichen Lizenzen. Alle Drittprodukte und -dienste, die Sie mit Produkten und Diensten von RIM erhalten, werden lediglich zu Ihrem Vorteil, ohne Mängelgewähr und ohne ausdrückliche oder konkludente Bedingungen, Billigungen, Garantien, Zusicherungen oder Gewährleistungen jeglicher Art von RIM bereitgestellt, und RIM übernimmt in diesem Zusammenhang keine Haftung. Die Verwendung von Drittprodukten und -diensten unterliegt Ihrer Zustimmung zu den Bedingungen separater Lizenzen und anderer geltender Vereinbarungen mit Dritten, sofern sie nicht ausdrücklich von einer Lizenz oder anderen Vereinbarung mit RIM behandelt wird.

Bestimmte in diesem Dokument angeführte Funktionen erfordern eine Mindestversion von BlackBerry® Enterprise Server, BlackBerry® Desktop Software und/oder BlackBerry® Device Software.

Die Nutzungsbedingungen für RIM-Produkte und -Dienste werden in einer entsprechenden separaten Lizenz oder anderen Vereinbarung mit RIM dargelegt. KEINE DER IN DIESER DOKUMENTATION DARGELEGTE BESTIMMUNGEN SETZEN IRGENDWELCHE AUSDRÜCKLICHEN SCHRIFTLICHEN VEREINBARUNGEN ODER GEWÄHRLEISTUNGEN VON RIM FÜR TEILE VON RIM-PRODUKTEN ODER -DIENSTEN AUSSER KRAFT.

Research In Motion Limited
295 Phillip Street
Waterloo, ON N2L 3W8
Canada

Research In Motion UK Limited
Centrum House
36 Station Road
Egham, Surrey TW20 9LF
United Kingdom

Veröffentlicht in Kanada